Diplomarbeit

Konzeption und prototypische Realisierung eines

License-Management-System

Michael Andriczka

Oktober 2004

Betreuung:

Prof. Dr. rer. nat. Andreas Lux

Fachbereich Design und Informatik
Fachhochschule Trier
University of Applied Sciences

FACHHOCHSCHULE TRIER
UNIVERSITY OF APPLIED SCIENCES
Fachbereich
DESIGN UND INFORMATIK

Autor: Michael Andriczka

Titel: Konzeption und prototypische Realisierung eines License-Management-System

Studiengang: Angewandte Informatik

Betreuung: Prof. Dr. rer. nat. Andreas Lux

Oktober 2004

Unterschrift des Autors

Danksagung

Mein Dank gilt allen Personen, die zum Gelingen der vorliegenden Diplomarbeit beigetragen haben.

Ganz besonders bedanke ich mich bei Herrn Prof. Dr. Andreas Lux, der mir die Realisierung diese Diplomarbeit ermöglichte und mir als Betreuer stets beratend zur Seite stand.

Weiterhin möchte ich Herrn Prof. Dr. Karl-Heinz Klösener für seine kompetenten Hinweise und seine sofortige Hilfsbereitschaft danken.

Ebenso danke ich Herrn Edwin Gerdon und Herrn Wolfgang Kästel, welche mir die Realisierung der Diplomarbeit bei DaimlerChrysler AG ermöglichten.

Abschließend möchte ich mich bei dem Team ITI/ETR-CAS für seine gute Unterstützung und die freundliche Aufnahme bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BANF	Bestellantrag der neueren Form
BMS	Bestands-Management-System
BS/IF-LCC	Bureau Services / Infrastructure Facility Management – Licence Competence Center
CLP	Contractual License Program
DBMS	Datenbank-Management-System
DDL	Data Definition Language
DML	Data Manipulation Language
ER-Modell	Entity-Relationsship-Modell
EULA	End User License Agreement
<fk>	Foreign key
HTML	Hypertext-MarkUp-Language
HTTP	Hypertext Transfer Protokol
IPS	International Procurement Services
ITC/TC-RM	Information Technology Commercial Vehicles / Trucks Consulting & Business Solutions Resourcen Management
ITI/ETR-CAS	Information Technology Infrastructure /European Trucks – Client- and Administration Services

JDBC	Java Database Connectivity
MES	Material-Einkaufs-System
MVC	Model-View-Controller
NF	Normalform
PA	Passport Advantange
PAX	Passport Advantange Express
<pi>	Primary identifier
<pk>	Primary key
RPC	Remote Procedure Call
RSVP	Relationship Suggested Volume Price
SEP	Symatec Elite Program
SQL	Structured Query Language
SVP	Symantec Value Program
TBYB	Try Before You Buy
TLP	Transactional License Program
WSAD	Websphere Studio Application Developer

1. Einleitung

Software ist heute für Unternehmen unverzichtbar und bietet diesen viele Möglichkeiten. Sie macht Unternehmen im täglichen Wettbewerb leistungsfähiger und produktiver. Doch um wettbewerbsfähig bleiben zu können, sollten alle Ressourcen dieser Software ausgenutzt werden und Unternehmen müssen mit den ständigen Veränderungen und dem schnell fortschreitenden Technologiewandel Schritt halten. Dafür benötigen sie immer häufiger neue Softwareprogramme oder Software-Updates, die von Unternehmen in Form von Lizenzen erworben werden. Durch den schnellen Wechsel und die vielen Änderungen in diesem Sektor sind eine genaue Planung des Softwareeinsatzes, eine exakte und effiziente Verwaltung der gekauften Softwarelizenzen unverzichtbar. Diese Kontrolle der bestehenden Ressourcen wird durch ein License-Management-System gewährleistet. Die Abteilung ITI/ETR-CAS des Unternehmens DaimlerChrysler am Standort Wörth zeigte großes Interesse an der Realisierung eines Prototyps, der diese Verwaltung von Softwarelizenzen zur Aufgabe hat. Die Konzeption und Realisierung eines Prototyps des License-Management-Systems ist das Ziel dieser Diplomarbeit. Es werden mit Hilfe des Prototyps Wege und Möglichkeiten aufgezeigt, wie mit Hilfe eines ausgereiften Systems ein effizientes Lizenz-Management betrieben werden kann, welches dem Unternehmen nicht nur Kosten spart, sondern auch zusätzliche Sicherheit bringt. Dabei zeigt nicht nur der Aspekt der Kostenersparnis die Aktualität dieses Themas, sondern auch die Sicherheit und die rechtliche Absicherung, die für ein Unternehmen unverzichtbar ist. Ziel dieser Diplomarbeit ist dabei nicht ein vollständiges Programm, sondern ein Prototyp, der die verschiedenen Aspekte aufzeigt die bei dieser Problematik zu beachten sind und Lösungsmöglichkeiten anbietet.

Um einen Prototypen entwickeln zu können, der auf die spezifischen Bedürfnisse der Lizenzverwaltung im Unternehmen DaimlerChrysler am Standort Wörth zugeschnitten ist, werden in *Kapitel 2* einige Lizenzierungsarten und herstellerspezifische Volumenlizenzverträge vorgestellt, die einen Einblick in die heutige Praxis des Lizenzeinkaufs geben.

In *Kapitel 3* werden die Beschaffungswege beschrieben, mittels derer DaimlerChrysler unternehmensweit Lizenzen einkauft. Es werden die Schwierigkeiten aufgezeigt, welche die geschilderten Beschaffungswege mit sich bringen.

Eine Analyse der Anforderungen, die an das License-Management-System gestellt werden, wird im *4. Kapitel* durchgeführt. Diese Analyse folgt drei Großbereichen: Es wird der Einsatzbereich und die Zielgruppe des Prototyps genau definiert, die Informationsanforderungen gegliedert und dargestellt, sowie die Funktionsanforderungen durch Unterteilung in kleinere Bereiche zugängig gemacht.

Auf der Grundlage der in Kapitel 4 ermittelten Informationen wird in *Kapitel 5* die Lizenzverwaltung in einem abstrakten Modell dargestellt. Für die Erstellung dieses Entity-Relationship-Modells wird die Datenbank-Modellierungssoftware „PowerDesigner Evaluationsversion 10.1.0“ verwendet, da sie eine einheitliche Form und Notation ermöglicht. Nach der Erläuterung einiger Grundlagen des Entity-Relationship-Modells wird es auf die spezifischen Anforderungen der Lizenzverwaltung des Unternehmens DaimlerChrysler am Standort Wörth angewendet.

Das entwickelte Entity-Relationship-Modell wird in *Kapitel 6* in ein relationales Datenbankschema umgesetzt. Dies geschieht auf den zunächst erläuterten Grundinformationen zum relationalen Datenmodell.

Kapitel 7 demonstriert beispielhaft, wie mit Hilfe des Model-View-Controller-Entwurfsmusters eine konkrete Funktion des License-Management-System umgesetzt wird.

Die Zusammenfassung des Geleisteten und der Ausblick auf weitere realisierbare Funktionalitäten des Prototyps bilden in *Kapitel 8* den Abschluss dieser Diplomarbeit.

2. Darstellung bestehender Lizenzierungsverträge

2.1. *Einleitung*

Unternehmen arbeiten mit sehr viel unterschiedlicher Software und müssen durch den ständig fortschreitenden Technologiewandel, die schnellen Veränderungen und Weiterentwicklungen in diesem Sektor in immer kürzeren Intervallen neue Softwareprogramme oder Software-Updates beschaffen. Deshalb ist eine genaue Planung des Softwareeinsatzes unumgänglich, besonders weil die Beschaffung von Software mit hohen Investitionen verbunden ist. Eine exakte und effiziente Verwaltung und folglich Kontrolle der gekauften Softwarelizenzen in Form eines sorgfältig durchdachten License-Management-Systems ist unverzichtbar und ermöglicht es den Unternehmen, den größten Nutzen aus diesen Investitionen zu ziehen.

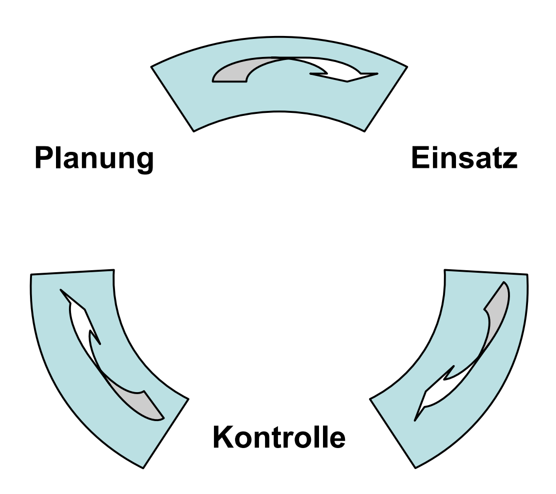


Abbildung 2-1: Kreislauf des optimalen Lizenzeinsatzes

Einkauf, Unterhaltung und Aktualisierung von Software ist sehr kostenintensiv. Umso wichtiger ist deswegen eine koordinierte Verwaltung dieser immateriellen Vermögenswerte des Unternehmens. Eine nachlässige Verwaltung dagegen führt nicht nur zu geringerer Produktivität und fehlender Effizienzsteigerung, sondern kann auch zum Auslöser für Softwarepiraterie werden.

Bei Softwarepiraterie werden nicht lizenzierte Softwarekopien genutzt, was nicht nur illegal ist und in Deutschland zivil- und strafrechtlich geahndet wird, sondern auch den Computer und die dort gespeicherten Daten beschädigen kann. Außerdem erhält der Anwender bei Raubkopien keine Service-Angebote für das Produkt, das heißt weder

eine Garantie, noch Wartungen, Benutzerhandbücher oder Upgrades. Die illegale Software kann eine veraltete Version sein, eine Testversion mit Programmfehlern oder eine fehlerhafte Kopie, die Daten beschädigt oder Viren enthält.
Als illegal gilt in diesem Bereich das nicht autorisierte Kopieren, Verbreiten oder der Verkauf von Software, es ist illegal, nicht autorisierte Softwarekopien im Internet zur Verfügung zu stellen, sie von dort herunter zuladen oder sie online zu verbreiten. „Jeder, der Software über die Lizenzbestimmungen hinaus kopiert oder raubkopierte Software einsetzt, verhält sich illegal und riskiert Verfolgung durch die Behörden – ob Fälscher, Händler oder Anwender, ob privat oder in Unternehmen.“ [bsa] Dabei kann nicht nur die Person, die gegen das Urheber- oder auch Patentrecht verstößt, zivil- und strafrechtlich angeklagt werden. Auch das Unternehmen, das diese illegalen Handlungen duldet, macht sich, wie [bsa] betont, strafbar, denn „Unternehmensleiter sind dafür verantwortlich, daß alle Software auf ihren Firmencomputern lizenziert ist.“ Manager und Verantwortliche vernachlässigen ihre Pflichten oft nicht vorsätzlich, doch die Komplexität der Materie erschwert es ihnen, den Überblick zu behalten wenn dem Unternehmen kein umfangreiches Programm zur Erfassung aller Lizenzen zur Verfügung steht. Neben den empfindlichen Strafen für die Nutzung von illegaler Software führt das Aufdecken eben dieser auch zu einem definitiven Imageverlust in der Öffentlichkeit.
So bietet, wie die Business Software Alliance [bsa] zusammenfasst, nur legale Software

- QUALITÄT
  Legale Software garantiert Qualität – von der Produktentwicklung über die Vollständigkeit der Lieferung bis hin zum Kundenservice.

- SICHERHEIT
  In die Neu- und Weiterentwicklung von Software werden tausende von Programmier- und Teststunden investiert, um sicherzustellen, daß sie zuverlässig läuft.

- DOKUMENTATION
  Kaufen Sie legale Software, erhalten Sie eine umfangreiche Bedienungsanleitung in Buchform oder als Onlinehilfe, kriminelle Händler liefern meistens keine oder unvollständige Dokumentation.

- SERVICE
  Als registrierter Kunde haben Sie ein Anrecht auf den technischen Kundendienst des Herstellers oder der von ihm hierzu autorisierten und geprüften Firmen. Bei illegaler Software helfen diese Firmen Ihnen nicht.

- UPGRADES
  Sind Sie beim Softwarehersteller als Kunde registriert, erhalten Sie besonders interessante Upgrade- und Erweiterungsangebote. Bei illegaler Software haben Sie kein Anrecht auf ein Upgrade. Sie müssen entweder eine legale Vollversion kaufen oder das Upgrade illegal kopieren, und riskieren damit Inkompatibilität.

Um das Anrecht auf diese Angebote zu bekommen, muss Software legal erworben werden, was über Lizenzen geschieht. Dabei ist eine Lizenz nach der Definition des [bsa] im Leitfaden für Softwaremanagement „Ein rechtsverbindlicher Vertrag, in dem die eine Partei der anderen bestimmte Rechte und Privilegien einräumt. Im Computerbereich erteilt ein Softwarehersteller einem Anwender üblicherweise das nicht exklusive, unbefristete Recht zur Anwendung seiner Software, untersagt jedoch weiteres Kopieren und Verteilen dieser Software an Dritte.“ Der Anwender benötigt die Genehmigung des Herstellers oder des Herausgebers von Computersoftware, um diese legal nutzen zu können. Diese Genehmigung wird normalerweise in Form eines Lizenzvertrages erteilt, durch den der Käufer jedoch nicht Eigentümer der Software wird, aber das eingeschränkte Recht erhält, das Programm entsprechend der im Lizenzvertrag festgelegten Bedingungen zu nutzen.
Wenn nun über die vorhandenen, gekauften und genutzten Lizenzen nicht genau Buch geführt wird, kann es sowohl zu Unterlizenzierungen kommen als auch zu Überlizenzierungen. Unterlizenzierung birgt die schon genannte Gefahr der strafrechtlichen Verfolgung, bei Überlizenzierung entsteht die Gefahr, dass das Unternehmen Mittel unnötig oder zur falschen Zeit investiert. Beides ist für das Unternehmen schädlich und oft mit hohen Kosten verbunden.
Um diesen Problemen vorzubeugen, wird das Lizenzmanagement eingesetzt. Es kontrolliert die Einhaltung des Nutzungsvertrages von installierter Software, indem es beispielsweise den nahenden Ablauf eines Vertrages anzeigt oder die nicht autorisierte Vergabe einer Lizenz an einen Computer. Es ermöglicht auch eine problemlose Verwaltung der Gesamtheit der erteilten Softwarelizenzen, da alle zur Verfügung stehenden Lizenzen in einer Übersicht dargestellt sind. Neben dem Schutz vor illegalen Handlungen ergibt sich so die Chance zur Kosteneinsparung. Bereits abgeschlossene Verträge bringen das Unternehmen in eine günstige

Verhandlungsposition gegenüber dem Softwareunternehmen und kann Preisnachlässe favorisieren.
Die angebotenen Lizenzverträge der Hersteller sind sehr unterschiedlich und eine genaue Evaluierung des Angebotes und ein gutes Management hilft hier, Kosten zu reduzieren. Es ergeben sich beispielsweise große Einsparungsmöglichkeiten durch die Optimierung der Beschaffungsprozesse. Grundsätzlich gilt, dass Kunden mit hohem Ordervolumen Lizenzen zu günstigeren Konditionen einkaufen, ähnlich einem Mengenrabatt. Viele große Hersteller wie Microsoft mit der Lizenzart SELECT bieten ihren Kunden an, Software für einen bestimmten Zeitraum zu evaluieren und diese erst nach gewisser, fest determinierter Zeit zu lizenzieren. Durch diese Demo-Lizenzen, wie sie auf Seite 4 beschrieben wird, kann das Unternehmen sicher gehen, dass es nur Lizenzen kauft, die wirklich und in dieser Form benötigt werden. Günstigere Konditionen werden von einigen Anbietern offeriert, wenn bereits bestimmte Lizenzen in der Vergangenheit von ihnen bezogen wurden. Für ein optimiertes License-Management-System ist es also unverzichtbar, abgelaufene und nicht mehr genutzte Softwarelizenzen zu archivieren, um sich so bei neuen Vertragsabschlüssen diesen Vorteil zunutze machen zu können.
Es ist also nötig, sich vor dem Einkauf und dem Abschluss von Lizenzverträgen genau über die jeweiligen Bedingungen und auch Möglichkeiten der Lizenzverträge zu informieren, weil nur dadurch die besten Bedingungen für das jeweilige Unternehmen geschaffen werden können.

2.2. Allgemein vorkommende Lizenzierungsarten

Bei der Betrachtung der verschiedenen Lizenzierungsarten ist evident, dass die von den großen Softwareherstellern angebotenen Lizenzarten sehr vielfältig und folglich sehr spezifisch sind. Das kommt der Befriedigung der individuellen Kundenbedürfnisse sehr entgegen. Zu dem Pool von Möglichkeiten, aus dem der Kunde wählen kann gehört, dass Lizenzen je nach Lizenzierungsart an ein bestimmtes Produkt, eine bestimmte Hardware oder aber an einen bestimmten Benutzer gebunden sein können.
Die Vielzahl der Softwarehersteller macht es nur schwer möglich, einen Überblick über die angebotenen Lizenzierungsarten zu bekommen. Dass sich hier ein sehr großes Spektrum eröffnet, zeigt sich an der folgenden Aufzählung, die nur einige Lizenzierungsarten beinhaltet. Eine nähere Ausführung dieser und weiterer Lizenzierungsarten ist im Anhang dargestellt.

- Demo (Evaluierungs-Lizenz)
- Floating (concurrent) in einem Netzwerk
- Upgrade Versionsnummer
- Post-use-payment
- Pay-per-use
- Netzwerk-Segmente (site licence)

Viele dieser Lizenzierungsarten werden auch im Unternehmen DaimlerChrysler verwendet und finden somit Eingang in das License-Management-System. Sie werden hier als individuelle Lizenzen gespeichert und ihren jeweiligen Besonderheiten, wie eine zeitliche oder nutzerspezifische Begrenzung, wird hier Rechnung getragen.
So wird bei der Eingabe der neu gekauften Lizenzen in das License-Management-System angegeben, mit welcher Vertragsart der Kauf der Lizenz vollzogen wurde. Wird bei der Ersterfassung beispielsweise der Wartungsvertrag Maintenance angegeben ist klar ersichtlich, dass der Käufer ein Recht darauf hat, innerhalb der Laufzeit des abgeschlossenen Lizenzvertrages die jeweils aktuellste Version zu installieren. Das License-Management-System verfügt darüber hinaus auch über ein Feld „Vertragsdefinition", das einen schnellen Überblick über die Bedingungen des Vertrags ermöglicht.
Somit werden viele der aufgeführten Lizenzierungsarten mit ihren spezifischen Nutzungsmöglichkeiten gekauft und das License-Management-System bietet Möglichkeiten, diese individuellen Bedingungen übersichtlich darzustellen, damit sie optimal genutzt werden können.

2.3. Herstellerspezifische Volumenlizenzmodelle

Nach der Darstellung der Lizenzierungsarten werden verschiedene Volumenlizenzierungsprogramme von großen und kleineren Herstellern betrachtet. Dadurch wird deutlich, wie wichtig eine genaue Auswahl der abzuschließenden Lizenzverträge ist und welchen großen finanziellen Vorteil der Käufer durch einen abgestimmten Lizenzeinkauf erreichen kann. Annähernd alle großen Software-Hersteller bieten bei dem Lizenzverkauf ein individuelles Konzept an, in welches sich der Kunde, je nach Größe und Kapazität seines Auftrages, einordnen kann.
Die Betrachtung der einzelnen Lizenzierungsverträge wird demnach vor allem mit Blick auf die diversen Rabattkonditionen vorgenommen, welche von den Herstellern

angeboten werden. Anhand dieser kann der Kunde bei sorgfältiger Planung und Auseinandersetzung mit der Thematik einen großen finanziellen Vorteil erreichen. Gleichzeitig kann sich der Käufer durch verschiedene Wartungsverträge wie eine Maintenance- oder Update-Lizenz die Unterstützung des Herstellers bei aufkommenden Problemen, sowie die jeweils neuesten Produkte sichern. Durch eine sorgfältige Auswertung des Lizenzbedarfs und eine genaue Verwaltung der aktiven und nicht mehr aktiven Lizenzen im License-Management-System können so die optimalen Bedingungen beim Abschluss eines Lizenzvertrags erreicht werden.

2.3.1. Adobe-Volumenlizenzmodell

Das Unternehmen Adobe bietet seinen Kunden für den Erwerb und die Verwaltung von Software-Lizenzen die Adobe Open Options-Lizenzprogramme [adobe] an. Dabei umfasst Adobe Open Options zwei unterschiedliche Programme: Das Transactional License Program (TLP) und das Contractual License Program (CLP), welche die unterschiedlichen Kundenprofile, Branchen und Produktanforderungen abdecken. Weiterhin besteht die Möglichkeit, für ein Adobe Open Options-Program einen Maintenance-Program (MP) als optionale Komponente zu erwerben. Dieses Programm stellt eine Art Technologie-Gewährleistung dar, welche den Kunden durch Upgrades über die neuesten Versionen der lizenzierten Adobe-Software informiert und ihm diese zur Verfügung stellt.
Der Preis der lizenzierten Produkte wird von Adobe durch ein Punktesystem berechnet, welches den anrechenbaren Wert jeder Produktlizenz bestimmt. Dabei wird die Anzahl der erworbenen Lizenzen mit der jeweiligen Punktanzahl des Produktes multipliziert und die Summe in eine Rabattstufe eingeteilt und berechnet.
Die Lizenzierung von Adobe-Software über Adobe Open Options-Programme bieten dabei mehrere Vorteile im Vergleich zu dem Erwerb von Einzellizenzen:

- niedrigere Lizenzkosten
- geringerer Auftrags- und Lizenzverwaltungsaufwand
- optimale Anforderung an individuelle betriebliche Anforderungen
- Upgrades im Rahmen des Maintenance-Program

Transactional License Program

Die Volumenlizenzierung mit dem Transactional License Program (TLP) richtet sich in erster Linie an kleinere und mittelständische Unternehmen. Es bietet die Möglichkeit, Lizenzen von Adobe-Software zu beziehen, ohne einen Vertrag zu unterschreiben oder eine Vorhersage über zukünftige Lizenzeinkäufe zu treffen. Das heißt, dass hier keine langfristigen Verpflichtungen aus Sicht des Kunden eingegangen werden müssen. Der Preis richtet sich dabei nach dem Umfang des TLP. Der Kunde erhält für jeden Kauf einer Produktlizenz oder die Registrierung einer Lizenz im Rahmen des Maintenance-Program eine bestimmte Anzahl an Punkten. Dabei muss für den Abschluss eines TLP eine Mindestpunktzahl von 5 erreicht werden. Die Anzahl der Lizenzen wird mit der individuellen Punktzahl der Lizenz multipliziert und in eine Rabattstufe eingeteilt. Der Kunde profitiert von diesen Rabattstufen, da er im Vergleich zu Einzellizenzen einen niedrigeren Preis zu entrichten hat.

Rabattstufe	Anzahl der Punkte
X	5 – 19
A	20 – 99
B	100 – 499
C	500 – 999
D	1000 – 4999
E	5000 – 19999
F	20000 und mehr

Tabelle 2-1: Rabattstufen des Transactional License Program

Contractual License Program

Beim Contractual License Program (CLP) handelt es sich um ein flexibles Volumenlizenzmodell für mittelständische bis große Unternehmen. Dieses Programm basiert auf der Vorhersage des voraussichtlichen Software-Lizenzbedarfs eines Unternehmens über 24 Monate hinweg. Aufgrund dieser Vorhersage wird eine einzige Preisstufe für alle Lizenzeinkäufe ermittelt. Dies ist ein weiterer Punkt, der für eine genaue Software-Verwaltung spricht. In einem optimalen License-Management-System kann über längere Zeiträume hinweg beobachtet werden, welche und wie viele Lizenzen eines bestimmten Produktes benötigt werden. Das CLP kombiniert hierbei Vergünstigungen durch einen Grosseinkauf mit der Möglichkeit, den Erwerb und dessen Kosten auf einen Zeitraum von 24 Monaten zu verteilen.

Kunden verpflichten sich mit dem Abschluss eines CLP-Vertrags über einen Zeitraum von 24 Monaten mindestens 1000 Punkte zu sammeln, wobei mindestens 10% der Punkte innerhalb der ersten 30 Tage nach Vertragsunterzeichnung und mindestens 40% der Punkte bis zum Ende des ersten Vertragsjahres bestellt werden müssen.
Der vom Kunden zu entrichtende Preis orientiert sich, wie auch beim TLP, an der Menge an Software-Lizenzen und dessen Punkte. Darüber hinaus erhöht sich die Punktzahl, wenn das Unternehmen neue Lizenzen erwirbt, bereits vorhandene Lizenzen erweitert oder eine Lizenz im Rahmen des Maintenance Program (MP) registriert.

Rabattstufe	Anzahl der Punkte
D	1000 – 4999
E	5000 – 19999
F	20000 und mehr

Tabelle 2-2: Rabattstufen des Contractual License Program

Maintenance Program

Das Maintenance Program (MP) von Adobe ist mit einem Upgrade-Gewährleistungsprogramm vergleichbar. Wenn ein Kunde ein MP für ein Produkt erwirbt, hat er Anspruch auf alle Verbesserungen, Fehlerbehebungen, Aktualisierungen und Upgrades. Durch die Registrierung der Lizenz als Maintenance-Vertrag im License-Management-Program ist klar ersichtlich, für welche Lizenzen diese Unterstützung des Herstellers genutzt werden kann.
Das MP wird als optionale Komponente zu TLP oder CLP angeboten. Beim Erwerb der oben genannten Leistungen im Rahmen des MP erhält der Kunde zusätzliche Punkte, die für TLP oder CLP angerechnet werden und sich auf die Höhe der verfügbaren Preisnachlässe auswirken. Auch kann ein MP für bereits erworbene Lizenzen bestellt werden, wodurch sich alle installierten Produkte einheitlich verwalten lassen.
Die Laufzeit eines MP für ein Software-Produkt läuft am Ende eines CLP-Vertrags beziehungsweise 24 Monate nach dem Bestelldatum einer TLP-Vereinbarung ab. Das heißt, wenn ein Kunde zum Beispiel ein Produkt am 21.10.2003 für eine Maintenance registriert, steht ihm bis einschließlich 21.10.2005 die vertraglich abgesicherte Unterstützung des Herstellers zu.

Die Lizenzprogramme von Adobe im tabellarischen Überblick:

Vorteil/Merkmal	Transactional License Program (TLP)	Contractual License Program (CLP)	Maintenance Program (MP)
Kundenprofil	Einzelfirmen und kleine Organisationen	Mittelständische bis Große Unternehmen/ Organisationen	Teilnehmer an Adobe Open Options
Niedrigere Software-Kosten	Ja	Ja	Ja
Einfachere Bestellung und Verwaltung	Ja	Ja	Ja
Mindestabnahme	5 Punkte	1.000 Punkte	wie Lizenzprogramm
Anzahl Rabattstufen	7	3	wie Lizenzprogramm
Zusage für weitere Aufträge	Keine	24 Monate	wie Lizenzprogramm
Web-basiertes Lizenzwerkzeug	Enthalten	Enthalten	Enthalten
Maintenance Program	Optional	Optional*	k.A.
Produktumfang	Umfassende Auswahl	Umfassende Auswahl	Enthalten
Gültigkeit	Innerhalb einer Region	Weltweit**	wie Lizenzprogramm
Vertriebsweg	Fachhändler	ALCs	ALCs

Tabelle 2-3: Lizenzprogramme von Adobe

* Die Anzahl der Bestellungen für das Maintenance-Program darf maximal der Anzahl der Lizenzen entsprechen, die nach dem sechsten Vertragsmonat bestellt wurden.
** Gilt auch für Tochtergesellschaften weltweit.
ALCs = Acrobat License Center
k.A. = keine Angabe

2.3.2. Symantec-Volumenlizenzmodell

Unter dem Namen Symantec Security License Program gestaltet das Unternehmen Symantec für Unternehmen und Behörden aller Größen unterschiedliche Lizenzvereinbarungen. Dabei werden zwei Volumenlizenzvereinbarungen unterschieden: das in [symantec] erläuterte Symantec Value Program (SVP), welches

für kleine und mittlere Unternehmen und Behörden ohne Niederlassungen in anderen Ländern empfohlen wird, sowie das Symantec Elite Program (SEP) [symantec] für mittlere und große Unternehmen und Behörden gleichwohl für multinationale Konzerne. Optional zu den beiden Lizenzprogrammen ist es möglich, eine Gold Maintenance für die bestehenden Produktlizenzen zu erwerben.

Symantec Value Program

Das Symantec Value Program bietet Unternehmen, die ihren Softwarebedarf bisher durch den Kauf einzelner Lizenzen für Softwarepakete abgedeckt haben, eine Alternative. Mit diesem Programm ist es möglich, ein so genanntes Media-Pack mit der gewünschten Anzahl der benötigten Softwarelizenzen zu kaufen. Daraufhin erhält der Kunde vom Unternehmen Symantec ein Lizenzzertifikat mit den für den Software-Lizenz-Kauf geltenden Lizenzbestimmungen und -bedingungen. Voraussetzung dafür ist eine Mindestabnahme einer Server-Lizenz oder von zehn Desktop-Lizenzen. Der Preis eines Lizenzzertifikats richtet sich hier an der Anzahl der Softwarelizenzen einzelner Produkte innerhalb einer Produktfamilie aus. Die Preiskategorien sind dabei in acht gestaffelten Rabattstufen strukturiert:

Rabattstufe	Anzahl der Server/ Anzahl der User/ Nodes
S	1+
A	10 – 24
B	25 – 49
C	50 – 99
D	100 – 249
E	250 – 499
F	500 – 999
G	1000+

Tabelle 2-4: Rabattstufen des Symantec Value Program (SVP)

Dabei richtet sich die Kategorie S an die Anzahl an Server und die Kategorien A – G an die Anzahl an User/Nodes.
Für den Einstieg in das Value Program oder zur Erreichung einer günstigeren Preiskategorie können die folgenden Lizenztypen miteinander kombiniert werden:

- Full License
- Upgrade License
- Crossgrade License
- Competitive Upgrade-License
- Maintenance
- Maintenance-Renewal

Symantec Elite Program

Das Symantec Elite Program (SEP) stellt für mittlere und große Unternehmen einen Abnahmevertrag mit einer 24-monatigen Laufzeit zur Verfügung, in dem unternehmensweite Bestellungen auch unterschiedlicher Symantec-Lösungen zusammengefasst werden können, um so die Vorteile höherer Mengenrabatte zu nutzen.
Im Rahmen des Elite Program bietet Symantec eine flexible Planung und Verwaltung der Software-Lizenzkäufe über einen bestimmten Zeitraum an. Dabei kann der Kunden zwischen zwei unterschiedlichen Kaufoptionen entscheiden:

- Commit-Option
  Eine Commit-Option stellt eine verbindliche Kaufverpflichtung innerhalb der 24-monatigen Vertragslaufzeit für den Kunden dar, um so die gültigen Mengenrabatte sofort bei Vertragsbeginn zu realisieren.

- Forecast-Option
  Die Forecast-Option stellt eine Minimierung der anfänglichen Lizenzausgaben dar. Bedingung dafür ist, dass der Kunde 25% seiner geplanten Lizenzeinkäufe über 24 Monate mit der Erstbestellung abdeckt.

Die oben genannten Kaufoptionen lassen sich in drei unterschiedliche Kategorien einteilen, welche sich anhand des Kaufpreises unterscheiden:

Kategorie	Euro
Minimum	85.000
Minimum	140.000
Minimum	198.000

Tabelle 2-5: Preiskategorien des Symantec Elite Program

Dabei gilt für die Commit-Option, dass Unternehmen eine Mindestbestellung in Höhe des unter Kategorie A angegebenen Mindestbestellwertes aufgeben.
Bei der Forecast-Option prognostiziert das jeweilige Unternehmen eine Abnahmemenge, deren Höhe mindestens dem unter Kategorie A angegebenen Mindestbestellwertes entspricht, und gibt dann eine Erstbestellung über 25% des voraussichtlichen Gesamtbestellwertes auf.

Symantec Gold Maintenance

Symantec bietet mit der Gold Maintenance einen Wartungsvertrag an, der individuell vom Kunden über 12 oder 24 Monate abgeschlossen werden kann. Weiterhin ist es auch möglich, den abgelaufenen 24-monatigen Maintenance-Vertrag wiederum als so genannten Renewal für 12 oder 24 Monate zu verlängern.
Im Rahmen dieses Maintenance-Vertrags erhält der Kunde für die zutreffenden Produkte die neuesten Upgrades, Sicherheits-Updates und einen Gold Support. Der Gold Support stellt einen Telefon-Support dar, der über die üblichen Geschäftszeiten für eine unlimitierte Anzahl an Anrufen für zwei definierte Kontakte zur Verfügung steht.

Beim Einkauf von Softwarelizenzen bieten sich dem Käufer auch bei diesem Anbieter vielfältige Möglichkeiten, Kosten einzusparen. Das ist möglich, wenn der Kauf genau kalkuliert wird und sich bisherige Erfahrungen bei der Anzahl der benötigten Lizenzen zunutze gemacht werden, was durch das License-Management-Program möglich wird.

2.3.3. IBM-Volumenlizenzmodell

Das Unternehmen IBM bietet für seine Kunden zwei unterschiedliche Volumenlizenzmodelle [ibm] an: Passport Advantage (PA) und Passport Advantage Express (PAX).
Beide Programme beziehen sich auf das gesamte Software-Portfolio von IBM, dessen Produktoptionen und Maintenance-Möglichkeiten.
Während Passport-Advantage sich an größere und multinationale Unternehmen richtet, ist das IBM-Programm Passport-Advantage Express auf die Bedürfnisse mittelständischer Unternehmen zugeschnitten. Bei jedem Kauf einer Software-Lizenz erhält der Kunde volle 12 Monate Software Maintenance. Er hat damit die Berechtigung, während dieses Zeitraumes die jeweils aktuellste Version der

erworbenen Software einzusetzen und Anspruch auf einen technischen Support, der mit dem Transaktionsdatum in Kraft tritt. Es ist möglich, dass Kunden ihre Software-Maintenance um 12 Monate verlängern oder bereits abgelaufene Software-Maintenance zu reaktivieren. Dies wird bei der Verwaltung der Lizenzen im License-Management-Program durch das „Archiv“ möglich, in welchem abgelaufene Lizenzen gespeichert werden und auch wieder zurück in den aktiven Lizenzpool gebracht werden können. Diese Reaktivierung (Renewable) ermöglicht Kunden mit abgelaufener Software-Maintenance, Software-Aktualisierungen und technischen Support für weitere 12 Monate zu erneuern, was allerdings zusätzliche Kosten im Vergleich zur fristgerechten Verlängerung der Software-Maintenance mit sich bringt.

Kunden erhalten besondere Preisstufen, solange die Software Maintenance erneuert wird oder weitere Käufe stattfinden. Finden in einem Zeitraum von 24 Monaten keine neuen Käufe statt und unterhält der Kunde keine aktive Software Maintenance mehr, wird der Vertrag beendet.

Die Preisbildung des Passport Advantage- und Passport Advantage Express-Programms basiert auf einem Punktesystem. Das bedeutet, dass jedes Produkt und jeder Service einem bestimmten Punktewert entspricht. Die Erstbestellung des Kunden wird in Punkte umgerechnet, mit deren Hilfe die Preisstufe des RSVP (Relationship Suggested Volume Price) bestimmt wird. Es gibt 8 Preisstufen, "BL" und "D" bis "J". Die Preisstufe "BL" ist nur für bereits existierende PA- und PAX-Kunden gedacht und ist keine Einstiegsstufe für neue Kunden. Die Stufen "I" und "J" sind nur auf besondere Anfrage und Genehmigung hin erhältlich.

Level	Punkte
BL	< 500
D	500 – 999
E	1.000 – 2.499
F	2.500 – 4.999
G	5.000 – 9.999
H	> 10.000
I	Unveröffentlicht
J	Unveröffentlicht

Tabelle 2-6: Punktekategorie des IBM-Volumenlizenzmodells

Passport Advantage

Das Passport Advantage-Volumenlizenzmodell ist zugeschnitten auf die Anforderungen größerer, multinationaler Unternehmen mit mehreren Standorten,

öffentlicher Institutionen und Behörden. Die vertragliche Vereinbarung muss hier in Form einer Beitrittserklärung abgegeben werden und bedingt für Neukunden einen Erstauftrag von mindestens 500 Punkten, wie in der folgenden Abbildung dargestellt wird:

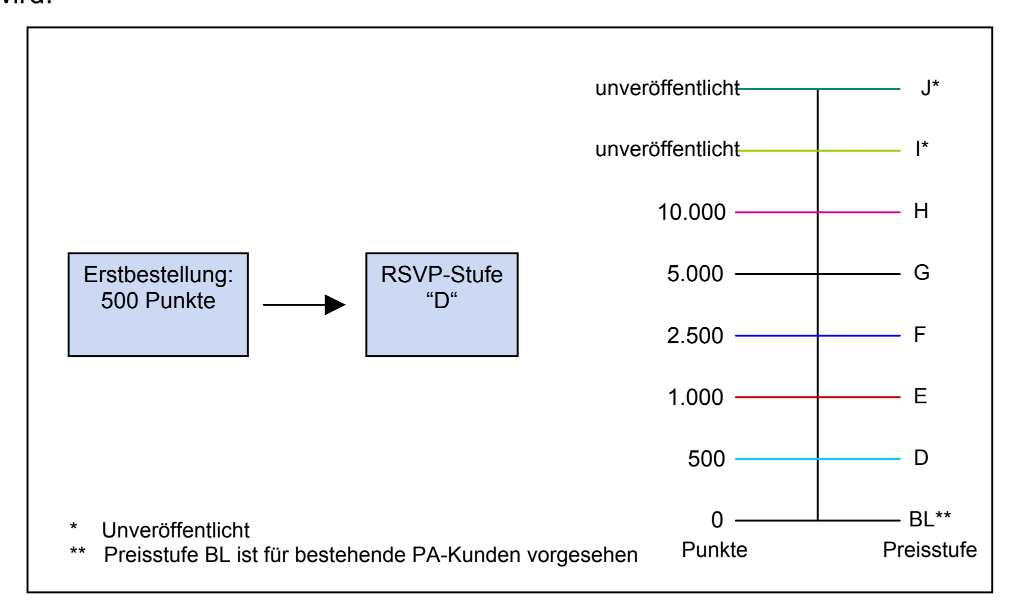


Abbildung 2-2: Erstbestellung im Rahmen des Passport Advantage-Programm

Die Preisbildung erfolgt allein nach dem „Relationship Suggested Volume Price Level" (RSVP). Der RSVP-Level wird nach jeder Transaktion neu kalkuliert und jährlich am so genannten „Anniversary Date" überprüft, wobei man bei entsprechenden Aufträgen in höhere beziehungsweise um höchstens ein Level niedriger eingestuft werden kann.

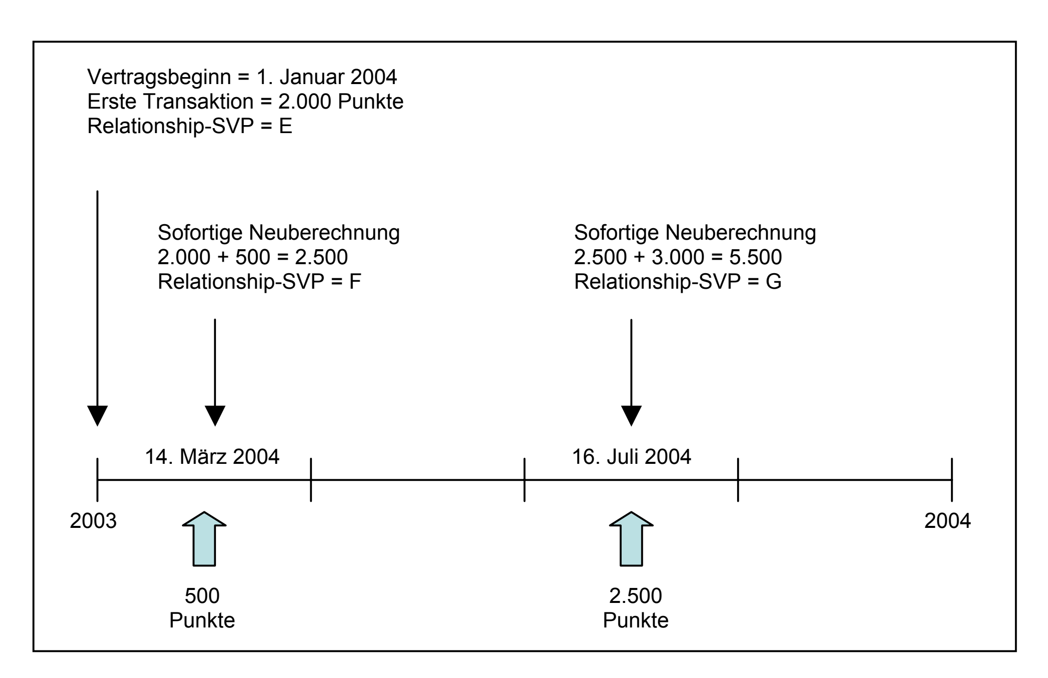


Abbildung 2-3: Darstellung der Passport Advantage Preiskategorie-Veränderung

In der Abbildung 2-3 vollzieht der Kunde am 1. Januar 2004 eine erste Transaktion für 2.000 Punkte. Damit ist die RSVP-Stufe des Kunden "E".
Am 14. März 2004 erwarb der Kunde zusätzliche Produkte im Wert von 500 Punkten. Die RSVP-Stufe für eine solche Transaktion über 500 Punkte ist die Preisstufe "D", die niedriger ist als "E", welche die aktuelle RSVP-Stufe des Kunden ist. Deshalb erhält der Kunde die günstigere Preisstufe "E" für diese Transaktion. Die Punktzahl dieser Transaktion wird sofort der Punktesumme des Kunden hinzugezählt, die sich damit auf 2.500 Punkte erhöht. Diese Summierung führt mit sich, dass die RSVP-Stufe des Kunden für den Rest der Laufzeit auf die Stufe "F" angehoben wird. Nachfolgende weitere Käufe und weitere Summierungen innerhalb derselben Laufzeit können die RSVP-Stufe noch mehr verbessern.
Am 16. Juli 2004 kauft der Kunde zusätzliche Produkte im Wert von 3.000 Punkten, die aufgrund des Umfangs dieser Transaktion nach der RSVP-Stufe "G" berechnet werden. Die sofortige Summierung der in diesem Jahr erlangten Gesamtpunktzahl für diese 3 Transaktionen ergibt 5.500 Punkte (2.000 Punkte + 500 Punkte + 3.000 Punkte). So befindet sich der Kunde nach dieser Transaktion für die restliche Laufzeit auf Stufe "G", sofern keine weiteren Transaktionen stattfinden, die den Kunden in eine günstigere RSVP-Stufe einordnen.

Passport Advantage Express

IBM bietet für kleine und mittelständische Unternehmen mit einem einzigen Standort ein einfaches Lizenzierungs- und Relationship-Modell namens „Passport Advantage Express“ (PAX) an.
Neukunden, die bei der Erstbestellung nicht die 500-Punkte-Grenze des Levels D erreichen, werden in das Level BL für PAX eingestuft und müssen somit beim Kauf einer Software-Lizenz im Vergleich zu PA keine vertragliche Vereinbarung eingehen. Die Software-Maintenance, das heißt unbegrenzter Support und Software-Aktualisierungen, gilt ab dem Tag des Lizenzerwerbs für 12 Monate. Ein Anniversary Date wie bei PA ist hier nicht möglich, da bei PAX jede Bestellung 12 Monate läuft und jede Bestellung ihr eigenes Anniversary Date hat, das jedoch wie bei PA nicht angeglichen werden kann.
Optional bietet PAX an, die Software-Maintenance für ein bereits lizenziertes Produkt um weitere volle 12 Monate zu verlängern.

Lizenzprogramme von IBM im tabellarischen Überblick:

Merkmal	Passport Advantage	Passport Advantage Express
Vertragliche Vereinbarung	Beitrittserklärung	Nicht erforderlich; Registrierung der Kundendaten
Minimale Punktzahl bei erster Bestellung eines neuen Kunden	500	Nicht erforderlich
Preislevel	BL D – H	Suggested Retail Price, BL
Anniversary Date	Jede Bestellung hat ihren eigenen Anniversary Date, kann über Anteilmäßigen Erwerb angepasst Werden	Jede Bestellung hat ihren eigenen Anniversary Date, eine Anpassung ist nicht möglich
Maintenance	Inklusive 12 Monate, anschließend Optional	Inklusive 12 Monate, anschließend Optional
Suggested Volume Pricing	RSVP	-
Anpassung des RSVP	Nach jeder Transaktion	-
RSVP-Abstufung/Jahrestag	Maximal ein Level	-

Tabelle 2-7: Lizenzprogramme des Unternehmens IBM

3. Organisatorische Rahmenbedingungen

3.1. Einleitung

Die Ist-Analyse untersucht die derzeitige Situation des Lizenz-Managements im Unternehmen DaimlerChrysler AG am Standort Wörth. Hierbei beschreibt die Ist-Analyse die momentane Aufgabenbearbeitung des täglichen Lizenzgeschäfts und deckt aktuelle Probleme auf.

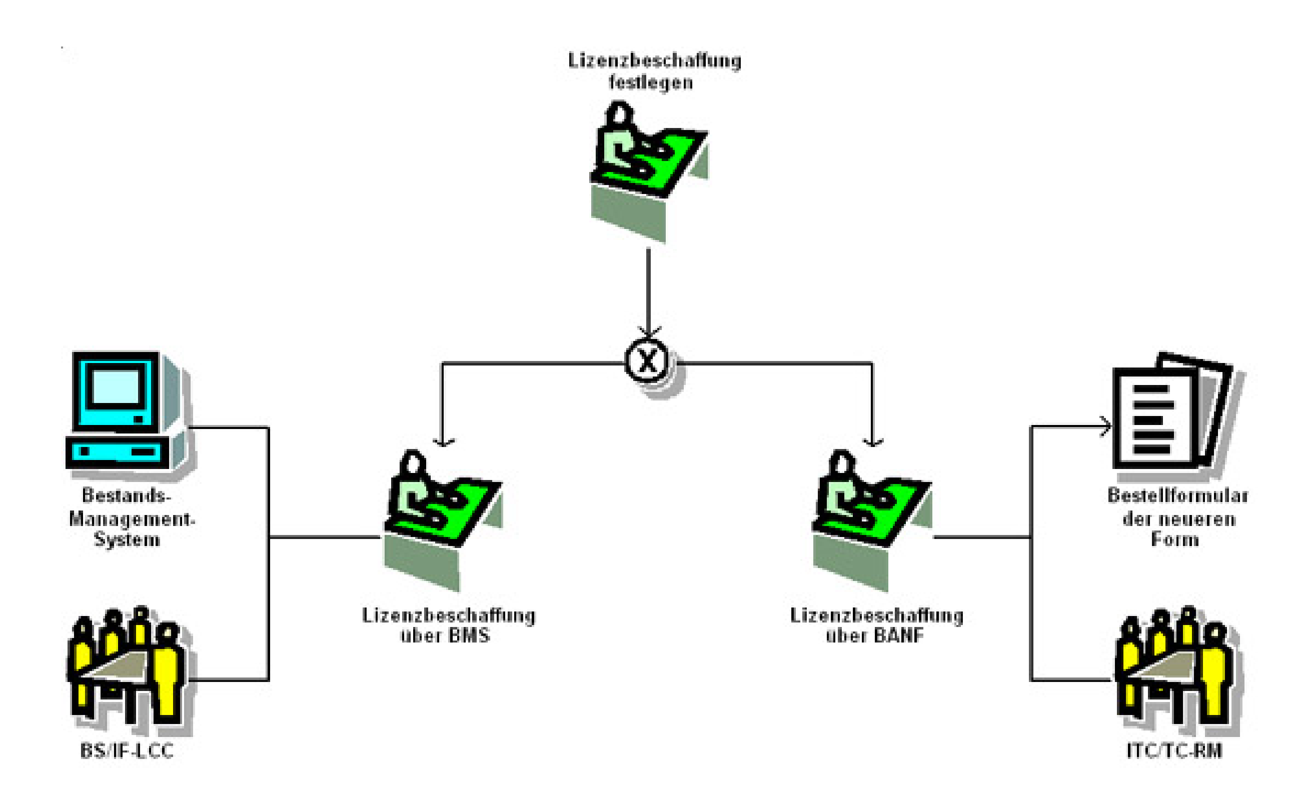


Abbildung 3-1: Darstellung des Office-Prozesses Lizenzbeschaffung

Die Vergabe von Software-Lizenzen wird durch einen Sachbearbeiter der Abteilung ITI/ETR-CAS vorgenommen. Hier gibt es zwei Möglichkeiten, die notwendigen Lizenzen zu beschaffen, siehe Abbildung 3-1.

Der jeweilige Sachbearbeiter kann die Software-Lizenzen mithilfe des Bestands-Management-Systems, kurz BMS genannt, über die Abteilung BS/IF-LCC beschaffen. Dieser Fall trifft zu, wenn es sich um eine strategische Software handelt, die unternehmensweit von DaimlerChrysler eingesetzt wird.

Die andere Möglichkeit ist die Beschaffung der Software-Lizenz über ein Bestellformular der neueren Form, kurz BANF genannt, durch die Abteilung ITC/TC-

RM. Diese Möglichkeit der Beschaffung trifft zu, wenn es sich um eine Spezialsoftware handelt, die individuell für einen Anwender beschafft wird.

3.2. Lizenzvergabe über BMS

3.2.1. Systembeschreibung

Im Unternehmen DaimlerChrysler AG am Standort Wörth wird für das Lizenz-Management das System BMS genutzt, welches von dem Unternehmen T-Systems als eine Individualsoftware für DaimlerChrysler erstellt wurde. Dieses System ist ein integriertes Bestands-Management-System, in dem die unterschiedlichen EDV-Aufträge organisiert werden, was auch das Lizenz-Management beinhaltet. In Abbildung 3-2 wird das Startmenü des BMS dargestellt, mit dem man die notwendigen Funktionen zur Auftragsbearbeitung aufrufen kann.

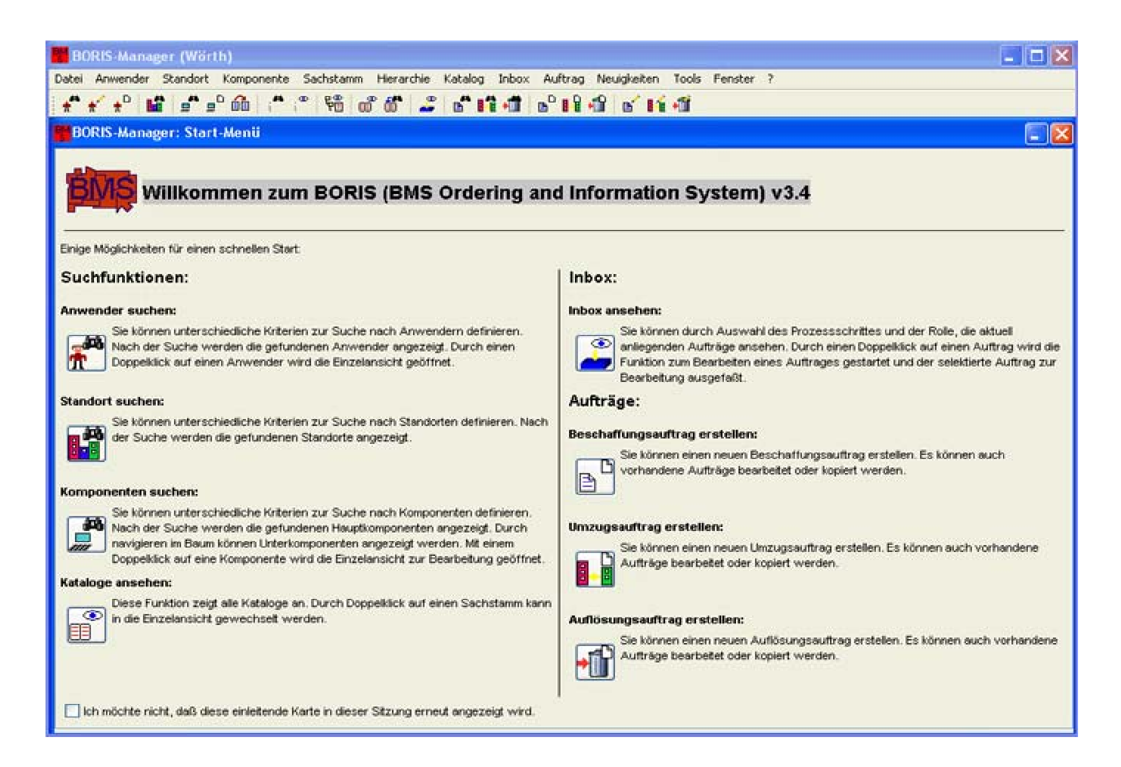


Abbildung 3-2: Darstellung des Bestands-Management-Systems

Folgende Aufträge werden über das BMS durch einen Sachbearbeiter vorgenommen:

- *Beschaffungsauftrag:*
  Beauftragung einer Neu-, Ersatz- bzw. Erweiterungsbeschaffung eines Geräts bzw. einer Software-Lizenz
- *Umzugsauftrag:*
  Festlegung eines neuen Standorts, einer neuen Abteilung oder eines neuen Anwenders
- *Auflösungsauftrag:*
  Rückgabe eines Geräts oder einer Software-Lizenz

Die Aufträge werden entsprechend den Vorgaben eines Workflow Schritt für Schritt bearbeitet. Über eine Statistik kann der ursprüngliche Antragsteller die Auftragsbearbeitung nachvollziehen und beispielsweise prüfen, ob Aufträge beispielsweise die zeitliche Begrenzung überschritten haben.
Jeder Auftrag kann in seinem aktuellen Prozessschritt im Workflow selektiert werden. Somit kann der Sachbearbeiter auf einen Blick die relevanten Aufträge auswählen und bearbeiten.
Eine Gruppe gleichartiger Aufträge kann man auch zu einem Sammelauftrag zusammenfassen, welcher dann wie ein Einzelauftrag durch den Workflow geleitet wird, vorausgesetzt, dass die Produkte nicht inventarisiert werden. Eine Inventarisierung von Software wird erst ab einem Betrag > 1004 Euro vorgenommen.
Nachdem eine Bestellung abgeschlossen ist, wird der Bestand des Endanwenders um die installierten Komponenten auf dem dazugehörigen Computer erhöht. Dieser Komponentenbestand kann über BMS verwaltet werden. Zusätzlich bietet BMS hierbei eine vollständige Anwender- und Standortverwaltung, die eine komplette Bestandsverwaltung erst ermöglicht.
Die Leistungsmerkmale und Nutzenpotenziale von BMS sind dementsprechend:

- komplette, workfloworientierte Beschaffungslogistik
- Bereitstellung einer einheitlichen Recherchemöglichkeit
- Bezug von Daten aus Quellsystemen
- maschinelle Datenversorgung von Folgesystemen
- vollständige Bestandsverwaltung bis zum Einzelarbeitsplatz

Der Großteil der Software-Lizenzen, die für DaimlerChrysler am Standort Wörth benötigt werden, wird mit einem Beschaffungsauftrag, der durch den jeweiligen

Workflow des BMS läuft, beschafft. Die bestellten Lizenzen werden über die Abteilung BS/IF-LCC, welche ihren Sitz in der Unternehmenszentrale von DaimlerChrysler AG in Stuttgart hat, bestellt und angeliefert.
Eine der Aufgaben des BS/IF-LCC besteht darin, unternehmensweit strategische Software für die Anwender des Unternehmens festzulegen und mit entsprechenden Softwarehäusern geeignete Lizenzmodelle auszuhandeln. Diese Software wird dann, wie in Abbildung 3-3 zu sehen, durch diese Abteilung in den Softwarebestand des BMS aufgenommen und für die jeweiligen Administratoren in den Standorten des Unternehmens DaimlerChrysler AG zur Verfügung gestellt. Aus diesem Softwarebestand können die Administratoren ihren Kunden, welche in unserem Fall die Mitarbeiter der jeweiligen Standorte von DaimlerChrysler AG sind, strategische Softwareprodukte anbieten.

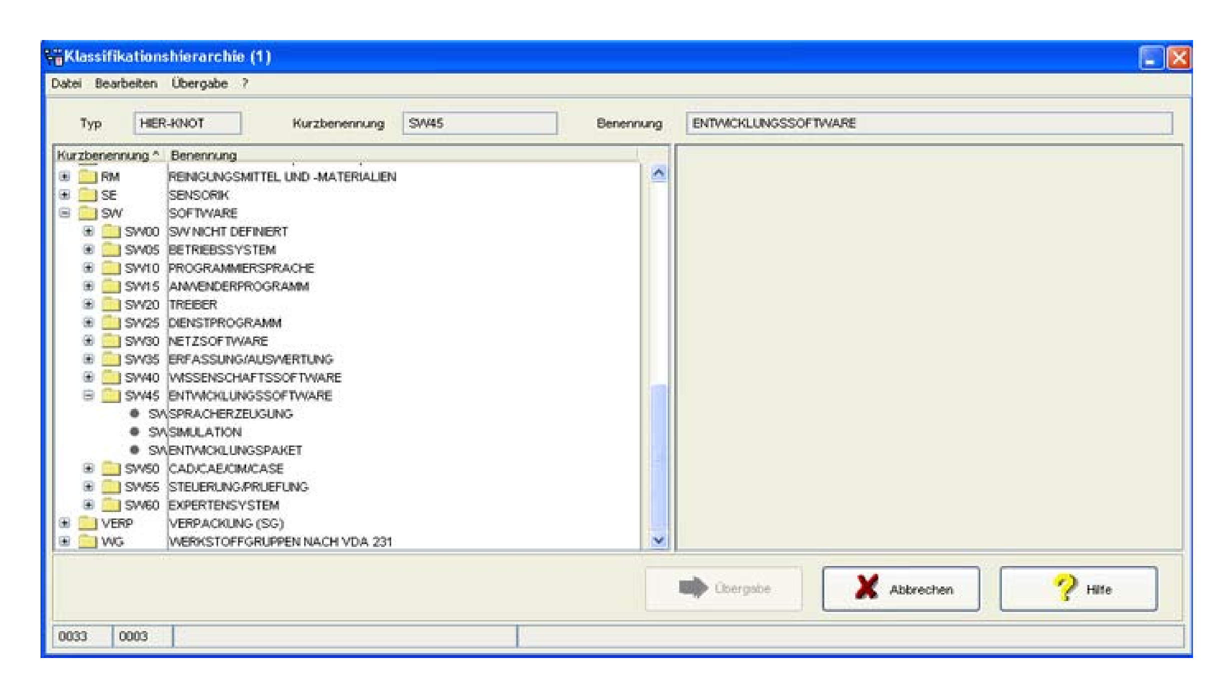


Abbildung 3-3: Darstellung der zur Verfügung stehenden Software im BMS

3.2.2. Verbesserungsmöglichkeiten zur Lizenzbeschaffung im BMS

Bei den weiteren Arbeitsschritten nach der Beschaffung der Lizenzen und somit der Vergabe der Software-Lizenzen innerhalb von DaimlerChrysler entsteht das Problem, dass eine Softwarelizenz mit ihrer spezifischen Sachnummer, hier QEV-Nummer genannt, als eine Zusatzkomponente des Computers behandelt wird. Diese QEV-Nummer hat die Softwarelizenz bei der Bestellung durch das BMS vom LCC zugewiesen bekommen, damit diese eindeutig identifizierbar ist.

Dadurch, dass diese QEV-Nummer als Attribut behandelt wird, also komplett auf den Computer des Endanwenders übergeht, ist sie nur auf diesem Computer dokumentiert. Daraus entsteht das Problem, dass bei einer möglichen Entsorgung des Computers auch die QEV-Nummer, welche die Software-Lizenz referenziert, gelöscht wird. Somit wird die von DaimlerChrysler AG gekaufte Lizenz vernichtet, auch wenn sie an anderer Stelle noch ohne weiteres genutzt werden könnte.
Dadurch entstehen für das Unternehmen nicht nur vermehrt Kosten bei der Neubeschaffung eben dieser Lizenzen, sondern es entsteht auch ein höherer Arbeits- und Zeitaufwand bei der Neubeschaffung und Vergabe der Lizenzen.
Diese mangelnde Dokumentierung der gekauften Lizenzen hat nicht nur in Verbindung mit der Entsorgung von Computern Nachteile.
Auch wenn angegeben ist, ob eine Lizenz vergeben ist oder nicht, stehen diese ungeordnet und einzeln aufgeführt in der Bildschirmmaske beieinander. Diese Tatsache erschwert den genauen und schnellen Überblick über den Lizenzbestand. Man kann somit nicht schnell erfassen, ob die benötigte Software noch zur Verfügung steht oder ob ihr gesamter Lizenzbestand vergeben ist.
In Abbildung 3-4 wird beispielhaft dargestellt, wie diverse Software-Lizenzen bei einem spezifischen Anwender oder in einem virtuellen Lager, kurz Lager060 genannt, vorhanden sind. Das heißt, dass sowohl vergebene als auch verfügbare Lizenzen zusammen in einer Bildschirmmaske dargestellt werden und somit der Überblick verloren geht und eine schnelle Handhabung erschwert wird.

Benennung ^	Bezeichnung	Abteilung	Kosten...	Name	Sachnummer
CIM	IDS PROF.SCHEER, ARIS-VKD VORGANGSKETTENDIAGRAMM	TOS/P60	5986	LAGER060	QEV111AAGA6X
CIM	IDS PROF.SCHEER, ARIS-VKD VORGANGSKETTENDIAGRAMM	TOS/P60	5986	LAGER060	QEV111AAGA6X
CIM	IDS PROF.SCHEER, ARIS-VKD VORGANGSKETTENDIAGRAMM	TOS/P60	5986	LAGER060	QEV111AAGA6X
CIM	IDS PROF.SCHEER, ARIS-VKD VORGANGSKETTENDIAGRAMM	TOS/P60	5986	LAGER060	QEV111AAGA6X
CIM	IDS PROF.SCHEER, ARIS V4.0 SERVER F. TOOLSET	TOS/P60	5986	LAGER060	QEV111AATHUP
CIM	IDS PROF.SCHEER, ARIS V4.0 SERVER F. TOOLSET	TOS/P60	5986	LAGER060	QEV111AATHUP
CIM	IDS PROF.SCHEER, ARIS-EASY DESIGN 4.0	TOS/P60	5986	LAGER060	QEV111AATHU2
CIM	IDS ARIS TOOLSET INKL. WARTUNG BIS JAHRESENDE	EG/E	3136	LEIPOLD	QEV111AA6R25
CIM	IDS ARIS WEB DESIGNER INKL. WARTUNG BIS JAHRESENDE	EG/AVW-KVS	3246	MACHINEK	QEV111ABYQYA
CIM	IDS ARIS TOOLSET INKL. WARTUNG BIS JAHRESENDE	ITC/TC	5436	MARTINEK	QEV111AA6R25
CIM	IDS PROF.SCHEER, ARIS-EASY DESIGN 4.0	P/PDC-E	9076	MORNHINWEG	QEV111AATHU2
CIM	IDS ARIS WEB DESIGNER INKL. WARTUNG BIS JAHRESENDE	ITC/TC	3246	NIELSEN	QEV111ABYQYA
CIM	IDS ARIS TOOLSET INKL. WARTUNG BIS JAHRESENDE	ITC/TC	5436	RADTKE	QEV111AA6R25
CIM	IDS ARIS TOOLSET INKL. WARTUNG BIS JAHRESENDE	LW/S	0426	SCHAEFER	QEV111AA6R25
CIM	IDS ARIS TOOLSET 4.0 BASISSYSTEM	VL/AK FZA	6446	SCHERRER	QEV111AAV250
CIM	IDS ARIS TOOLSET 4.0 BASISSYSTEM	PF/MP	3286	SCHUDRA	QEV111AAV250
CIM	IDS ARIS TOOLSET INKL. WARTUNG BIS JAHRESENDE	ITC/TC	5436	SCHUELER	QEV111AA6R25
CIM	IDS ARIS WEB DESIGNER INKL. WARTUNG BIS JAHRESENDE	EG/E	3246	WERLING	QEV111ABYQYA
CIM	IDS ARIS SERVER INKL. WARTUNG 1 - 25 USER	ITC/TO	5986	WILKEN	QEV111AA6R27

Abbildung 3-4: Darstellung der Lizenzen im BMS anhand des Suchbegriffs „ARIS“

Ein weiteres Problem besteht darin, dass kein Warnsystem vorhanden ist, das den Sachbearbeiter frühzeitig informiert, wenn Lizenzen auslaufen oder bald keine dieser

Lizenzen mehr zur Verfügung stehen, also neue Lizenzen eines speziellen Produkts nachbestellt werden müssten, um einen reibungslosen Arbeitsablauf zu gewährleisten.

Dadurch, dass kein umfassender und genauer Überblick über die Lizenzen gegeben ist, kann auch kein koordinierter Einkauf von Lizenzen erfolgen. Es ist von Vorteil, wenn nachvollziehbar wird, wie viele Lizenzen der einzelnen Produkte für die Endanwender benötigt werden und wie viele noch für den Gebrauch zur Verfügung stehen. Ist dies nicht klar ersichtlich, kommt es nicht nur zu doppelten Einkäufen, sondern es entstehen auch weitere Nachteile, wie:

- Produktivitätsverlust beim Anwender
- Finanzieller Verlust für das Unternehmen
- Mehraufwand in Form von Nachbestellung

Wenn dieser Überblick also nicht gegeben ist, kann auch keine ausführliche Analyse in Form eines Reports im BMS vorgenommen werden. In diesem Report kann dann festgestellt werden,

- welche Lizenzen wie oft genutzt wurden,
- welche Software-Lizenzen im Werk Wörth eingesetzt werden oder
- wie hoch die jährlichen Software-Beschaffungskosten sind.

Da diese Funktionalität, wie oben erwähnt, im BMS nicht gegeben ist, kann man aus ihr auch keine Rückschlüsse auf das weitere Einkaufsverhalten von Software-Lizenzen schließen. Man kann sich nicht an der bisherigen Nutzung von Software für weitere Einkäufe orientieren und somit oft benötigte Lizenzen in größerer Anzahl beschaffen. Wenn andere, weniger oft benötigte Lizenzbestände zuerst aufgebraucht beziehungsweise anderen Standorten des Unternehmens zur Verfügung gestellt werden, bevor neue Lizenzen gekauft werden, können in diesem Bereich Einsparungen vorgenommen werden.

Die fehlende Funktionalität zur Aufnahme von nicht strategischer Software in das BMS ist ein weiteres Problem, wenn also Spezialsoftware, welche über einen Beschaffungsauftrag der neueren Form besorgt worden ist, in das BMS aufgenommen werden soll. Diese fehlende Funktionalität führt zu dem Problem, dass der Datenbestand des BMS eine Inkonsistenz gegenüber dem gesamten Datenbestand des Unternehmens DaimlerChrysler AG am Standort Wörth aufweist. Dadurch ist es dem jeweiligen Sachbearbeiter auch nicht möglich, die über BANF beschafften

Software-Lizenzen entsprechend zu aktualisieren und den Anwendern zur Verfügung zu stellen.

3.3. Lizenzvergabe über BANF

3.3.1. Ablaufbeschreibung

Nicht alle Software-Lizenzen, die für Daimler Chrysler am Standort Wörth benötigt werden, werden über BMS beschafft. Handelt es sich bei der zu beschaffenden Software nicht um Strategiesoftware, welche von der Unternehmenszentrale DaimlerChrysler AG in Stuttgart vorgegeben wird, sondern um eine Spezialsoftware, erfolgt die Beschaffung dieser Software-Lizenz über eine BANF und nicht über das BMS.

BANF ist ein allgemein festgelegter Prozess, um einen benötigten Artikel beliebiger Art zu beschaffen. Hierbei muss der jeweilige Sachbearbeiter nach Zustimmung seines Vorgesetzten, welcher mindestens den Rang eines Teamleiters (E4) begleiten muss, einen BANF mittels eines BANF-Bestellformulars, siehe dazu Abbildung 3-5, erstellen. In der Abteilung ITC/TC-RM werden die Daten durch einen zuständigen Sachbearbeiter im SAP R/3-System erfasst und eine spezifische BANF-Nummer generiert. Nach dieser Datenerfassung geht zum einen das BANF-Bestellformular an die auftragstellende Abteilung mit der neuen BANF-Nummer zurück und des Weiteren wird die Bestellung durch das SAP R/3-System automatisch an das Material-Einkaufs-System der Abteilung IPS weitergeleitet, welche für den Einkauf im Werk Wörth verantwortlich ist. Diese Abteilung beschafft den notwendigen Artikel, beispielsweise eine Software-Lizenz, und liefert diesen der Abteilung ITI/ETR-CAS zu.

Von dieser Abteilung wird die angelieferte Software-Lizenz an die auftragstellende Abteilung ausgeliefert.

Die entsprechende Abteilung, welche die Beschaffung der Software-Lizenz beauftragt hat, muss die Kosten für die Beschaffung der Software-Lizenz aus ihrem abteilungseigenen Budget übernehmen.

Bestellanforderung - Konzept

Belegart: ZNB / ZSAB (Kontr.-Nr. 20000..) / ZST / ZPR...

Anforderer, IT: ____________ Abteilung: ____________

Telefon: ____________ Datum: ____________

Antragsteller, IT: ____________ Telefon: ____________

E4

Kopftext/Verwendung: ____________

Vorg.-BANF: ____________

Jahresbanf ☐

Pos.	Pos.Text (Kurztext, max. 40 Stellen/Zeile)	Menge/ ME-Einh./St/LE	Waren-/ Ge.Eing.	BwrtPreis/Euro pro Einheit
		2		10

20
Gesamtsumme

Papieranlage ja / nein

Lieferant ____________

über PP1000? ja / nein

Lief.-Nr. ____________ Coins-Titel ____________

Lieferdatum ____________ zu entl. Kst ____________

Lagerort ____________

Bezeichnung ____________

Kontierung ____________

CO-Auftr.-Nr. 60230000......

PSP-Element 060A/I

Sonstige Bemerkungen für Lieferant/Einkauf ____________

BANF-BELEG-NUMMER: ____________

(M.-Nr. intern)

Für Lieferant und Rechn. Anerkennung nur die spätere Best.-Nr. 106000................ verwenden !

Abbildung 3-5: Darstellung eines BANF-Bestellformulars

3.3.2. Verbesserungsmöglichkeiten zur Lizenzbeschaffung im BANF

Diese über eine BANF beschafften Software-Lizenzen besitzen keine QEV-Nummer und sind somit auch nicht im BMS-Datenbestand aufgeführt, was zur Inkonsistenz des gesamten Datenbestands von DaimlerChrysler AG am Standort Wörth führt. Es ist also nicht klar ersichtlich, welche Lizenzen neben der strategischen Software für die einzelnen Abteilungen eingekauft wurden und somit zur Verfügung stehen. So kann ein

neu entwickeltes Lizenzsystem auch diese über BANF beschafften Lizenzen in den Datenbestand mit aufnehmen und gewährt so eine bessere Übersicht über die gekauften Lizenzen.

3.4. Zusammenfassung der organisatorischen Rahmenbedingungen

Das Lizenz-Management optimal durchzuführen ist eine wichtige Aufgabe, da nur so der Lizenzbestand optimal ausgenutzt werden kann und dadurch dem Betrieb den größtmöglichen Nutzen bringt. Die Analyse der momentan eingesetzten Beschaffungswege, welche zur täglichen Arbeit des Lizenzgeschäftes genutzt werden, zeigen deutliche Ansatzpunkte zur Verbesserung auf. Die aufgeführten Möglichkeiten zur Software-Lizenz-Beschaffung sind verbesserungsfähig. Die kurz skizzierten Vorschläge zur Optimierung werden im folgenden Kapitel 4 „Anforderungsanalyse des License-Management-System“ detailliert aufgeführt und näher erläutert. Es werden hier Wege und praktische Vorschläge gefunden, wie das momentane Lizenzmanagement optimiert werden kann.

4. Anforderungsanalyse des License-Management-System

4.1. Einleitung

Nachdem nun ein Einblick in die unterschiedlichen Lizenzarten gegeben wurde und verschiedene Volumenlizenzverträge beispielhaft dargestellt und die organisatorischen Rahmenbedingungen geklärt wurden, folgt an dieser Stelle die Anforderungsanalyse an das zukünftige Datenbanksystem. Hier werden die Aspekte der Benutzergruppen, des Anwendungsbereiches, die zu speichernden Informationen als auch die Systemfunktionalitäten im tieferen Detail analysiert. Die Anforderungen, die dabei an das License-Management-System gestellt werden, sind sehr vielfältig. Diese Analyse der so genannten Realwelt ist für den weiteren Verlauf des Datenbankentwurfs und damit für die Transformation in ein entsprechendes Datenmodell von entscheidender Bedeutung, da die weiteren Entwurfschritte auf dieser Anforderungsanalyse aufbauen.
Im Folgenden wird zuerst die Benutzergruppe wie auch der Anwendungsbereich des License-Management-System beschrieben. Anschließend werden die Informationsanforderungen an das System gesammelt und abschließend die Funktionsanforderungen, welche das System erfüllen muss, dargestellt. Wesentlich ist, dass die Daten, welche die zu beschreibenden Objekte definieren, im Datenbanksystem gespeichert und ihre Beziehungen zueinander verdeutlicht werden.

4.2. Zielgruppe und Einsatzbereich

Das License-Management-System dient der Verwaltung von Software-Lizenzen des Unternehmens DaimlerChrysler AG am Standort Wörth. Zielgruppe des Systems sind die Mitarbeiter der Abteilung ITI/ETR-CAS, welche unter anderem für die Beschaffung und Vergabe von Software-Lizenzen zuständig sind. Mithilfe des License-Management-System ist es den Mitarbeitern möglich, einen aktuellen Überblick über den Lizenzbestand des Werks Wörth zu erhalten, diese von zentraler Stelle aus zu verwalten, also zu vergeben, zurückzunehmen und zu aktualisieren.

4.3. Informationsanforderungen

Bevor die Funktionsanforderungen beschrieben werden können, ist es notwendig, eine Analyse der statischen Informationen durchzuführen, die für das License-Management-System bei DaimlerChrysler am Standort Wörth benutzt werden sollen.
Dabei zeigen sich drei Kernbereiche, die im Folgenden einzeln beschrieben werden, auch wenn die einzelnen Bereiche miteinander in Verbindung stehen:

- Kernbereich Vertrag
- Kernbereich Produkt
- Kernbereich Computer

4.3.1. Kernbereich Vertrag

Der erste Bereich umfasst das Objekt „Vertrag", wobei hier die definitorischen Bestandteile des Vertrags, seine beschreibenden Attribute und die unterschiedlichen Vertragsausprägungen genannt werden.
Ein Vertrag ist im Lizenz-Management ein Rechtsgeschäft, das aus zwei übereinstimmenden, mit Bezug aufeinander abgegebenen Willenserklärungen besteht. Im Lizenz-Management bedeutet der Begriff „Vertrag", dass es sich um einen Lizenzvertrag handelt, der dem Endanwender das Nutzungsrecht eines Softwareprodukts einräumt. Die Vertragsinhalte sowie die Nutzungsbedingungen (EULA = End User License Agreement) variieren dabei von Hersteller zu Hersteller und von Vertrag zu Vertrag. Eine detaillierte Beschreibung unterschiedlicher Lizenz- als auch Paketverträge ist im Kapitel 2. „Darstellung bestehender Lizenzierungsverträge" gegeben.
Ein Vertrag wird im License-Management-System eindeutig durch seine Vertragsnummer definiert und kann mit dieser aus dem Datenbestand des License-Management-System identifiziert werden. Ein Vertrag wird immer für ein oder mehrere Lizenzen abgeschlossen.
Der Vertrag als individuelles Exemplar wird durch seine Vertragsart beschrieben, wie zum Beispiel eine End-User-License oder eine Maintenance-License und gehört zu einer Vertragskategorie, die entweder eine Softwarevollversion, eine Softwarelizenz oder eine Wartungslizenz sein kann.
Weiterhin besitzt ein Vertrag einen festgesetzten Vertragsbeginn und eine eindeutig definierte Laufzeit. Ferner kann ein Vertrag über eine bestimmte Nutzungsanzahl abgeschlossen werden.

Der Einzelpreis eines Produkts kann immer unterschiedlich sein, da viele Hersteller eigene Rabattkonditionen für bestimmte Produkte geben. Auf herstellerspezifische Konditionen wurde bereits in Kapitel 2.3. „Herstellerspezifische Volumenlizenzmodelle“ näher eingegangen.
Wird ein Vertrag, wie beschrieben, über einen Lieferanten beschafft, der über das Bestands-Management-System (BMS) oder den Bestellantrag der neueren Form (BANF) erfolgen kann, wird diesem Vertrag eine eindeutige Sachnummer zugewiesen. Diese Sachnummer kann eine QEV-Nummer oder eine BANF-Nummer sein und kann bis zum Bestellantrag zurückverfolgt werden. Im Gegensatz zur Vertragsnummer, die lediglich zur internen Bearbeitung der Daten genutzt wird, steht die Sachnummer dem Anwender zur Vertragsidentifizierung zur Verfügung.

4.3.2. Kernbereich Produkt

Der zweite Bereich „Produkt“ bezieht sich auf die zu nutzende Software, die auf einen Computer und nicht wie im bisherigen Lizenz-Management bei DaimlerChrysler AG am Standort Wörth auf einen Anwender dokumentiert wird. Jedes Produkt unterliegt eindeutigen Vertragsbedingungen, die vom Endanwender, der in unserem Fall der Lizenzmanager der Abteilung ITI/ETR-CAS ist, eingehalten werden müssen.
Das individuelle Produkt wird intern eindeutig durch seine spezifische Produktnummer identifiziert, ist aber vom Anwender durch seine Sachnummer eindeutig auffindbar. Diese Sachnummer ist identisch mit der Sachnummer des Vertrags, der über dieses Produkt abgeschlossen wurde. Dabei kann ein Vertrag auch mehrere Produkte beinhalten, die einerseits alle durch die gleiche Sachnummer identifiziert werden, andererseits individuelle Produktnummern besitzen.
Ferner wird ein Produkt durch seinen Hersteller und eine Produktbezeichnung beschrieben, welche in verschiedenen Versionen, Betriebssystemvoraussetzungen und Anwendersprachen beim Endanwender vorliegen können. Um ein Produkt nutzen zu können, muss dieses mittels eines Schlüssels frei geschaltet werden, wenn die Freischaltung nicht direkt durch den Hersteller erfolgt. Dieser Schlüssel kann in unterschiedlichen Arten vorliegen, wie beispielsweise in Form eines Dongles oder eines Lizenz-Key, der vom License-Management-System verwaltet wird. Dabei kann ein Lizenz-Key in unterschiedliche Kategorien, wie Einzelkey, Goldenkey oder Concernkey differenziert werden. Der Einzelkey ist ein Freischalt-Code für ein einzelnes Produkt, während der Goldenkey eine n-malige Freischaltung eines Produkts erlaubt. Der Concernkey wird von Unternehmen erworben, wenn ein Produkt in einer nicht definierbaren Anzahl eingesetzt und keine Schlüsseleingabe benötigt wird.

4.3.3. Kernbereich Computer

Der letzte Bereich „Computer“ beschreibt die Vergabe eines Produkts an einen Computer, welches auf einen Computer und nicht auf einen Anwender dokumentiert wird. Der spezifische Computer wird im License-Management-System durch seinen eindeutigen Computernamen identifiziert. Eine weitere Möglichkeit, einen Computer zu identifizieren, ist seine Inventarnummer, welche bei einer Neuanschaffung eines Gegenstandes vom Werk Wörth vergeben wird und durch welche der Lizenzmanager im Bestands-Management-System weitere Informationen, wie den Computernamen oder den Endanwender, ermitteln kann. Der Computer gehört einer LVM-Klasse an, die in PC-Workstation, PC-Notebook und Server unterschieden wird.
Ein weiteres beschreibendes Attribut eines Computers ist die Anzahl seiner Prozessoren. Dies ist notwendig, da sich der Lizenzvertrag auf die Anzahl der eingesetzten Prozessoren, wie zum Beispiel bei Server-Lizenzen, beziehen kann.

4.4. Funktionsanforderungen

Um die notwendigen Anforderungen an die Systemfunktionalität abzudecken, ist es notwendig, das License-Management-System in unterschiedliche Bereiche einzuteilen. Diese Bereiche sind:

- Vertrag
- Produkt
- Computer
- Report
- Stammdaten
- Archiv
- Vertragspool

4.4.1. Funktionsanforderung an den Vertrag

Wegen des ständigen Wechsels und den Änderungen der Lizenzmodelle unterschiedlicher Softwareanbieter wurde die Funktion „Vertrag“, welche verschiedene Ausprägungen des Contract-Managements darstellt, in das License-Management-System integriert. Diese Funktion bietet mit ihren verschiedenen Unterfunktionen

Möglichkeiten zur Verwaltung und Pflege der im Unternehmen DaimlerChrysler AG am Standort Wörth eingesetzten Lizenz- und Softwareverträge.
Die Funktion „Vertrag hinzufügen“ wird benötigt, wenn ein Vertrag für eine Software abgeschlossen wurde und dieser in den Datenbestand des License-Management-System aufgenommen werden muss. Der eingegebene Vertrag mit seinen dazugehörigen Produkten wird im Vertragspool aufgezeigt. Dabei können auch nachträglich beschaffte Produkte zu diesem Vertrag hinzugefügt werden.
Die Funktion „Vertrag entfernen“ wird benötigt, damit ein Vertrag aus dem Datenbestand des License-Management-System entfernt werden kann. Wird der Vertrag mit all seinen Produkten aus dem aktiven Datenbestand des License-Management-System entfernt, wird dieser zur Aufbewahrung im Archiv des Systems gespeichert. Weiterhin ist es möglich, auch einzelne Produkte eines Vertrags aus dem Datenbestand in das Archiv zu verlagern.
Die Funktion „Vertrag suchen“ ist notwendig, wenn sich der Lizenzmanager über die genaueren Details eines Vertrags informieren will. Durch diese Funktion wird der Vertrag durch Eingabe eindeutiger Schlüsselfelder aus dem Datenbestand des License-Management-System identifiziert und angezeigt.

4.4.2. Funktionsanforderung an das Produkt

Durch den Abschluss eines Vertrags wird dem Anwender die Nutzung bestimmter Produkte erlaubt. Diese Produkte sind vom Vertrag und seinen Bedingungen abhängig und werden durch dessen Sachnummer definiert.
Die Funktion „Produkt“ weist mehrere Unterfunktionen auf. Diese ermöglichen es, einen Überblick über die Lizenzen mit ihren dazugehörigen Produkten zu behalten. Weitere Funktionen sind die Vergabe spezifischer Produkte, deren Nutzungsrecht durch Lizenzen gekauft wurde, an Neugeräte oder Umzugsgeräte, Produkte von nicht mehr benötigten Altgeräten oder Umzugsgeräten entgegenzunehmen und veraltete Produkte aus dem License-Management-System zu entfernen. Außerdem ist es möglich, frühzeitig zu erkennen, wenn neue Lizenzbeschaffungen getätigt werden müssen. Diese Vorausschau gewährleistet einen kontinuierlichen Beschaffungsfluss und schützt vor unerwarteten Beschaffungsproblemen.
Die Funktion „Produkt zuweisen“ ermöglicht die Vergabe eines lizenzierten Produkts an einen Computer. Dabei wird der Bestand des vorhandenen Vertrags um die jeweilige Anzahl verringert. Ist für ein spezielles Produkt eine Freischaltung in Form eines Lizenzschlüssels vorgesehen, muss der Lizenzschlüssel gleichfalls auf den Computer dokumentiert werden.

Die Funktion „Produkt zurücknehmen“ ist für die Zurückbuchung der Lizenzen in den Vertragspool zuständig. Diese Funktion tritt in Kraft, wenn ein Computer entsorgt oder wenn dieser vom Anwender nicht mehr benötigt wird. Bei der Zurückbuchung der Lizenz wird der Bestand der jeweiligen Lizenzart im Produktpool um die Anzahl der Zurückbuchungen erhöht.
Die Funktion „Produkt suchen“ ermöglicht es, ein spezielles Produkt aus dem Datenbestande des License-Management-System zu identifizieren und mit seinen Attributen anzuzeigen. Diese Funktion ist notwendig, wenn ein Produkt beispielsweise an einen Computer vergeben wird oder Informationen zu einem bestimmten Produkt benötigt werden.

4.4.3. Funktionsanforderung an den Computer

Aus Gründen der Effizienz und Wirtschaftlichkeit werden die beschafften Software-Lizenzen von DaimlerChrysler im Werk Wörth immer auf einen speziell dazugehörigen Computer dokumentiert. Dies ist insbesondere wegen des ständigen Wechsels der Tätigkeitsgebiete der jeweiligen Anwender sinnvoll.
Die Funktion „Computer“ garantiert durch seine Unterfunktionen im License-Management-System eine kontinuierliche Aktualität des Lizenzbestands und vermeidet dadurch das Problem des Kontrollverlustes bezüglich der Überlizenzierung und eine rechtliche Absicherung gegenüber Unterlizenzierung. Mit den entsprechenden Unterfunktionen ist es möglich, dem Datenbestand des License-Management-System Computer hinzuzufügen, diese von dort zu entfernen und aus dem Datenbestand den gesuchten Computer zu identifizieren. Weiterhin ist es möglich, die auf dem Computer installierten Produkte zurück in den Vertragspool zu buchen, wodurch der Bestand um die jeweilige Anzahl erhöht wird.
Die Funktion „Computer hinzufügen“ wird benötigt, wenn ein Computer neu beschafft wird und noch nicht im Datenbestand des License-Management-System vorhanden ist. Bei diesem neu hinzugefügten Computer ist der Lizenzmanager in der Lage, Produkte zu vergeben, die anschließend auf den Computer dokumentiert werden.
Die Funktion „Computer entfernen“ ist besonders hilfreich, wenn ein bereits bestehender Computer entfernt werden muss und die vergebenen Lizenzen für die Softwareversion zurück in den Vertragspool eingelagert werden sollen, um so wieder dem aktiven Datenbestand zur Verfügung zu stehen. Dabei ist es wichtig, dass der Bestand der verfügbaren Lizenzen für das jeweilige Softwareprodukt um die entsprechende Anzahl der zurückgebuchten Lizenzen erhöht wird.

Die Funktion „Computer suchen“ wird genutzt, um einen bestimmten Computer aus dem Datenbestand des License-Management-System zu identifizieren, um beispielsweise diesen mit einer neuen Lizenz für ein Produkt auszurüsten oder eine Lizenz für ein installiertes Produkt in den Vertragspool zurück zuschreiben. Dazu ist es notwendig, dass der gesamte Bestand der auf diesem Computer installierten Produkte angezeigt wird.

4.4.4. Funktionsanforderung an den Report

Um einen aktuellen Überblick zu gewährleisten, kann dem License-Management-System eine Reportfunktionalität implementiert werden. Zum einen werden mithilfe der Reportdarstellung die auslaufenden Verträge und die Verträge, deren Bestand unterschritten wird, dargestellt und zum anderen ist es mit der Funktion „Report“ möglich, individuelle Reports zu erstellen und diese auf die Bedürfnisse von Marketing, Vertrieb und Logistik genau abzustimmen und auszugeben.

4.4.5. Funktionsanforderung an die Stammdaten

Mithilfe der Funktion Stammdaten ist es möglich, alle Stammdaten des aktiven Datenbestands des License-Management-System zu verwalten und zu pflegen. Nach einmaliger Eingabe neuer Stammdaten stehen diese dem Anwender zur Verfügung und sichern so die Konsistenz dieser Daten. Dabei werden zur besseren Übersicht einzelne Stammdatenkategorien unterschieden, wie zum Beispiel Vertragsart oder Betriebssystem.
Änderungen an den Stammdatenkategorien können durch das Hinzufügen neuer Daten in den Datenbestand oder das Entfernen nicht mehr benötigter Daten aus diesem Datenbestand vorgenommen werden. Dabei kann jedoch nur ein Datensatz aus dem Stammdatenbestand entfernt werden, wenn dieser nicht im aktiven Datenbestand verwendet wird.

4.4.6. Funktionsanforderung an das Archiv

Alle Verträge, die aus dem aktiven Datenbestand entfernt werden, kommen automatisch in das Archiv. Diese Funktion bewahrt somit die nicht mehr aktiv genutzten Lizenzverträge auf und stellt dadurch sicher, dass diese Daten gegebenenfalls reaktiviert oder für spätere Vertragsverhandlungen genutzt werden können.

4.4.7. Funktionsanforderung an den Vertragspool

Die Funktion Vertragspool stellt gleichzeitig die Startseite des License-Management-System dar. Auf dieser Seite werden alle aktiven Verträge mit ihren jeweiligen Produkten aufgeführt. Ein Warnsystem benachrichtigt dabei den Anwender, wenn der Bestand eines Vertrags unterschritten wird. Bei jeder Vergabe eines Produkts wird der Bestand des jeweiligen Vertrags um die entsprechende Anzahl verringert und äquivalent dazu bei jeder Zurücknahme erhöht.

5. Der konzeptionelle Datenbankentwurf

5.1. Einleitung

Das Ziel des konzeptionellen Datenbankentwurfs ist es, ein Teilausschnitt der realen Welt, hier die Lizenzverwaltung, in ein entsprechendes abstraktes Modell der zu modellierenden Miniwelt zu transformieren. Das Ergebnis soll ein zielsystemunabhängiges Datenbankschema sein, welches gegenüber einem Datenbank-Management-System, kurz DBMS, unabhängig ist.
Das Modell sollte nach [elm02] folgende Merkmale enthalten:

1. *Aussagekräftigkeit*: Das Datenmodell sollte aussagekräftig sein, damit unterschiedliche Datentypen, Beziehungen und Einschränkungen unterschieden werden können.

2. *Einfachheit und Verständlichkeit*: Das Modell sollte so einfach wie möglich sein, sodass auch nicht spezialisierte Benutzer es verstehen und seine Konzepte nutzen können.

3. *Minimalität*: Das Modell sollte eine kleine Anzahl von Basiskonzepten umfassen, deren Bedeutung eindeutig und nicht überlappend ist.

4. *Darstellung in Diagrammform*: Das Modell sollte über eine Notation für die Darstellung eines konzeptuellen Schemas in Form eines Diagramms verfügen, das sich leicht interpretieren lässt.

5. *Formalität*: Ein im Datenmodell ausgedrücktes konzeptionelles Schema muss eine formale eindeutige Spezifikation der Daten darstellen. Folglich müssen die Modellkonzepte genau und eindeutig definiert werden.

Um diese Merkmale umzusetzen, wird für die konzeptionelle Modellierung am häufigsten das Entity-Relationship-Modell (ER-Modell) eingesetzt, das von P. P. Chen im Jahre 1976 [che76] entwickelt wurde. Seit der Entwicklung dieser Modellierungsmethode wurde das ER-Modell mehrfach verbessert und erweitert.
Für die Entwicklung des konzeptionellen Datenmodells dieser Diplomarbeit wird die Datenbank-Modellierungssoftware „PowerDesigner Evaluationsversion 10.1.0“ des

Unternehmens Sybase verwendet. Mit dieser Software ist es dem Entwickler möglich, das konzeptionelle und das relationale Datenmodell in einer einheitlichen Form und Notation abzubilden.

Das ER-Modell basiert nach [heu00] auf drei Grundkonzepten:

1. Entitäten als zu modellierende Informationseinheiten

2. Relationships zur Modellierung von Beziehungen zwischen Entities

3. Attribute als Eigenschaften von einer Entität oder einer Beziehung

Mithilfe des ER-Modells ist es dem Entwickler möglich, das konzeptionelle Schema als eine kompakte Beschreibung der Datenanforderungen der Anwender darzustellen.
Dabei können die Entitätstypen, die Beziehungen sowie deren Attribute in ausführlicher Form grafisch dargestellt werden.
Im Rahmen dieser Diplomarbeit werden die grundlegenden Elemente des Entity-Relationship-Modells erläutert. Eine detaillierte Beschreibung des ER-Modells würde den Rahmen dieser Diplomarbeit überschreiten, für weiterführende Vertiefungen sei jedoch beispielsweise auf [Heu00] verwiesen.

5.2. Grundlagen des Entity-Relationship-Modells

5.2.1. Entität, Entitätsmenge, Entitätstyp und Schlüssel

Eine Entität (engl. entity) ist ein individuelles Exemplar von Elementen der realen oder der Vorstellungswelt. Dies können zum Beispiel Personen, Gegenstände oder Begriffe sein, über die Informationen gespeichert werden und die sich durch Attribute beziehungsweise Attributwerte voneinander unterscheiden. Eine Entitätsmenge (engl. entity set) ist eine Sammlung von gleichartigen Entitäten, das heißt von Entitäten mit den gleichen Attributen, aber unterschiedlichen Attributwerten. Für die Modellierung sind jedoch Entitätstypen (engl. entity type) relevant. Dabei gehören gleichartige Entitäten, also Elemente, die sich durch äquivalente Attribute beschreiben lassen, zu einem Entitätstyp.

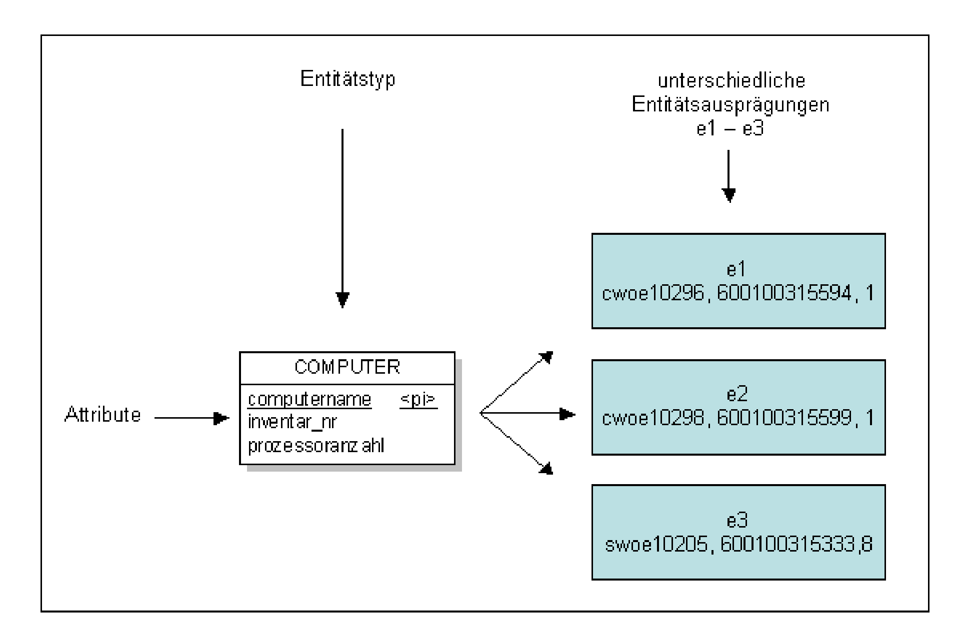


Abbildung 5-1: Darstellung Entität, Entitätsmenge, Entitätstyp und Schlüssel

Um eine Entität eindeutig aus der Entitätsmenge zu identifizieren, muss der Entitätstyp mit einem Schlüsselattribut versehen werden, den man auch Primärschlüssel (engl. primary key) nennt. Im Modellierungstool „PowerDesigner“ des Unternehmens Sybase wird das Primärschlüsselattribut auch als „primary identifier“ bezeichnet und durch die Zeichenfolge <pi> folgend auf ein entsprechendes Attribut unterstrichen dargestellt, siehe Abbildung 5-2. Die Suche nach dem am besten geeigneten Schlüsselattribut beziehungsweise den Schlüsselattributen geschieht nach dem Minimalitätsprinzip, das heißt, es soll die geringste Anzahl an Attributen zur Schlüsselbildung verwendet werden. Hier unterscheiden sich Entitäten mit atomaren Schlüsseln, die durch ein einziges Schlüsselattribut identifiziert werden können und Entitäten mit zusammengesetzten Schlüsseln, die eine Kombination verschiedener Attribute darstellen, welche die Entität eindeutig identifizieren. Alle anderen eindeutig identifizierenden Schlüsselattribute, welche nicht den Primärschlüssel bilden, sind Kandidatenschlüssel und dienen der späteren Vergabe von Indizes. Weitere Details zu Schlüsselarten werden im relationalen Datenmodell näher erläutert.

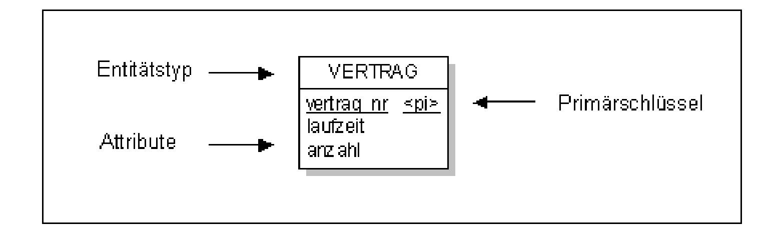


Abbildung 5-2: Primärschlüssel und Attribute des Entitätstyps „VERTRAG"

5.2.2. Beziehung, Beziehungsmenge und –typ, Rolle und Kardinalität

Zwischen Entitätstypen bestehen Abhängigkeiten, die durch Beziehungen (engl. relationship) ausgedrückt werden. Eine Beziehungsmenge (engl. relationship set) ist eine Sammlung von Beziehungen gleicher Art zur Verknüpfung von Entitätsmengen. Analog zum Entitätstyp stellt ein Beziehungstyp (engl. relationship type) die Abstraktion gleichartiger Beziehungen dar. In der Regel stehen zwei Entitätstypen in Wechselwirkung, es können aber auch mehrere assoziiert werden. Nach [elm02] spielt jeder Entitätstyp, der an einem Beziehungstyp teilnimmt, in der Beziehung eine bestimmte Rolle. Der Rollenname bezeichnet diese Rolle, die eine teilnehmende Entität des Entitätstyps in jeder Beziehungsinstanz spielt und vereinfacht die Erklärung, welche Bedeutung die Beziehung hat. Diese Beziehungen sind beispielhaft in Abbildung 5-3 dargestellt.

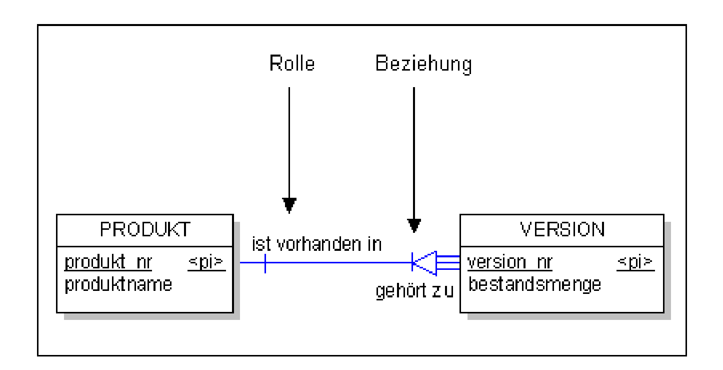


Abbildung 5-3: Beziehung zwischen zwei Entitäten

Die Kardinalitäten legen fest, wie viele Entitäten aus zwei unterschiedlichen Entitätsmengen (Entity_1, Entity_2) in Beziehung stehen können. In Abbildung 5-3 werden die daraus resultierenden Kardinalitäten in der Krähenfuß-Notation dargestellt. Die Beschreibung der Beziehungen werden in den folgenden Tabellen dargestellt. Die erste Tabelle stellt die „One-to-many"-Beziehungen dar. Hierbei werden die unterschiedlichen Beziehungen, wie in Tabelle 5-1 dargestellt, in „kann" (optional) und „muss" (mandatory) eingeteilt.

	Mandatory	One	Must exist one and only one
	Mandatory	Many	Must exist one or more
	Optional	One	May exist one or none
	Optional	Many	May exist one, more or none

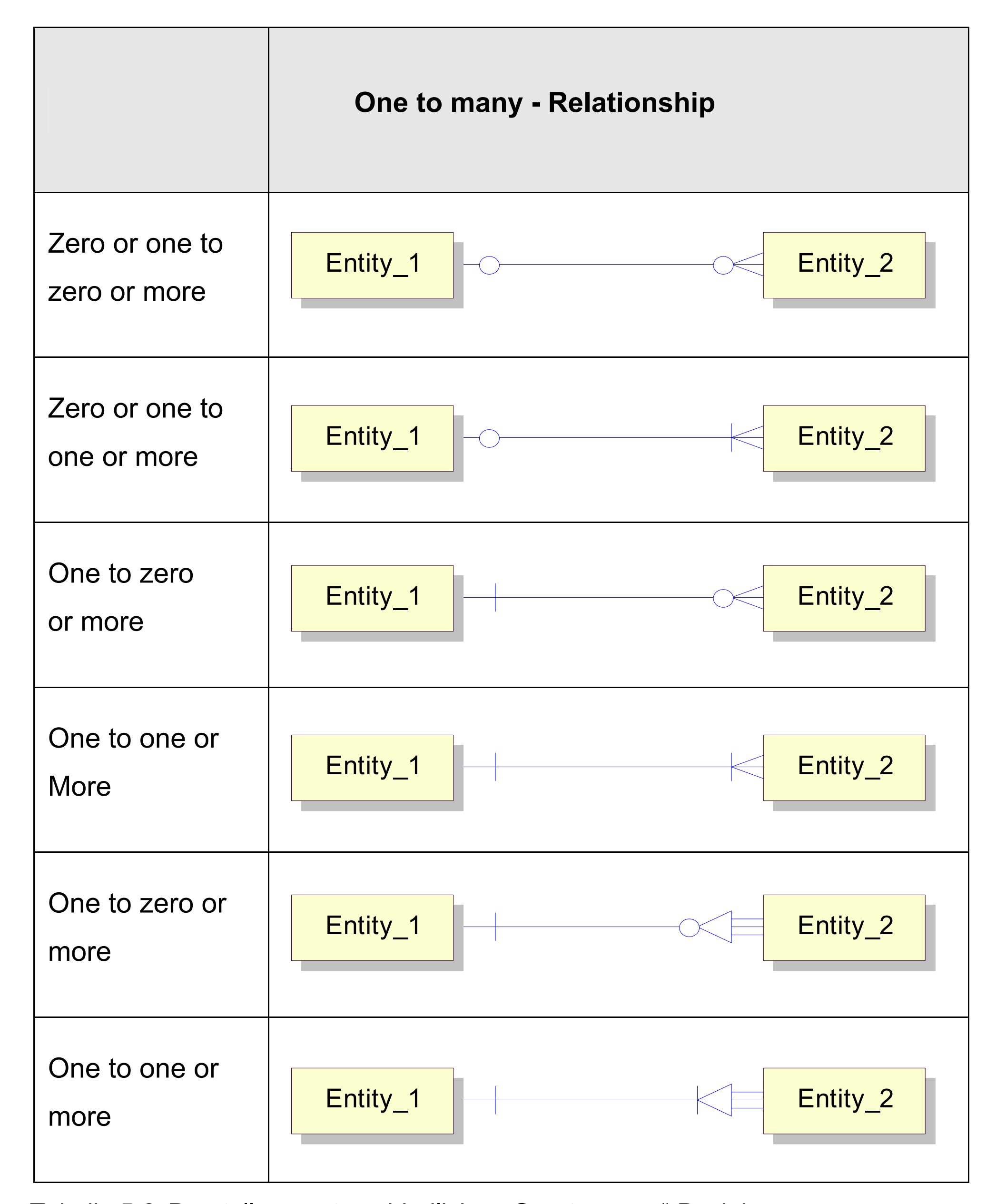
Tabelle 5-1: „muss"- und „kann"-Kardinalitätsverhältnisse

Die Beziehungen bedingen sich gegenseitig, was vor allem bei der Erklärung der "dependent"-Beziehungen wichtig ist. Bei diesen Beziehungen werden die Primärschlüssel als zusätzlich identifizierende Attribute an die abhängige Entität vergeben. Diese schwachen Entitäten sind von der übergeordneten Entität abhängig und oft nur in Kombination mit dem Schlüssel der übergeordneten Entität eindeutig identifizierbar. Hier gibt es unterschiedliche Kardinalitätsabhängigkeiten zwischen den in Beziehung stehenden Entitäten, wie in der folgenden Tabelle 5-2 dargestellt ist.

	Dependent	One	Must depend on one
	Dependent	Many	Must depend on one or more
	Dependent	One	Must depend on one or none
	Dependent	Many	Must depend on one, more or none

Tabelle 5-2: abhängige Kardinalitätsverhältnisse

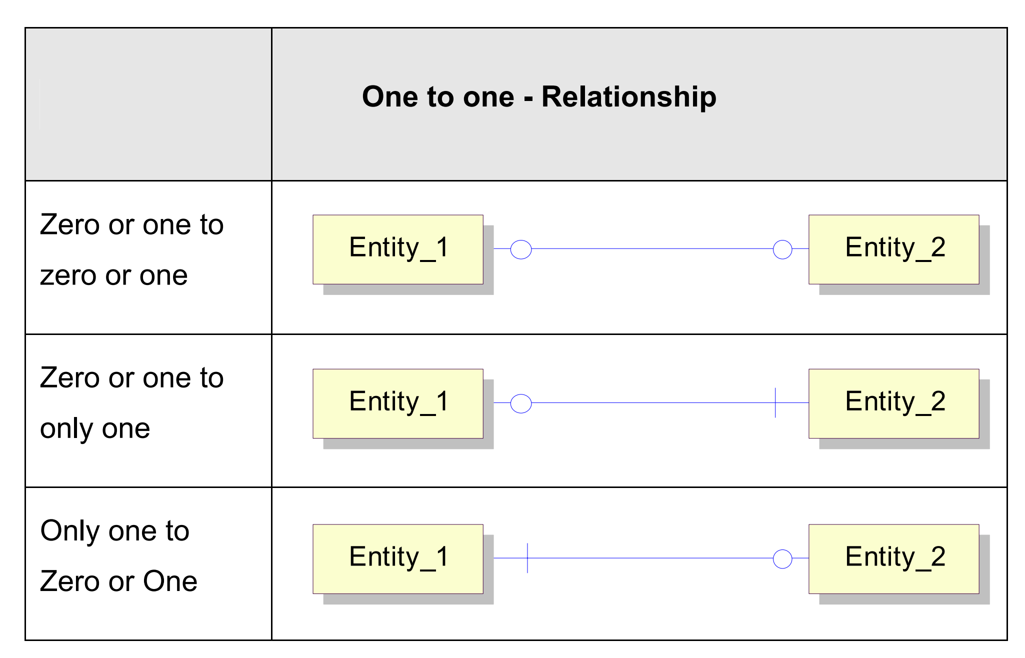
Die Beschreibung der „One-to-many"-Beziehungen und der verwendeten Notation wird in Tabelle 5-3 näher erläutert.



	One to many - Relationship
Zero or one to zero or more	Entity_1 Entity_2
Zero or one to one or more	Entity_1 Entity_2
One to zero or more	Entity_1 Entity_2
One to one or More	Entity_1 Entity_2
One to zero or more	Entity_1 Entity_2
One to one or more	Entity_1 Entity_2

Tabelle 5-3: Darstellung unterschiedlicher „One-to-many"-Beziehungen

Falls einer Entität im Gegensatz zur „One-to-many"-Beziehungen nur eine oder keine Entität zugeordnet werden kann, spricht man von einer „One-to-one"-Beziehung. Die verschiedenen Ausprägungen dieser 1:1-Beziehung sind in der folgenden Tabelle dargestellt.



	One to one - Relationship
Zero or one to zero or one	Entity_1 ○————○ Entity_2
Zero or one to only one	Entity_1 ○————| Entity_2
Only one to Zero or One	Entity_1 |————○ Entity_2

Tabelle 5-4: Darstellung unterschiedlicher „One-to-one“-Beziehungen

5.2.3. Attribute und Attributtypen

Ein oder mehrere Attribute, auch Eigenschaften genannt, dienen der Beschreibung von Entitäten sowie der Beschreibung der Beziehungen zwischen den einzelnen Entitäten. Jedes Attribut besitzt dabei für eine bestimmte Entität oder eine bestimmte Beziehung einen individuellen Wert. Identifiziert ein atomares oder auch ein zusammengesetztes Attribut eine Entität, wird dieses auch als Schlüsselattribut bezeichnet. Weitere Details zu Schlüsselarten und Schlüsselbildung sind im relationalen Modell beschrieben.

In einem ER-Modell können verschiedene Attributtypen vorkommen:

- einfache (atomare) und zusammengesetzte Attribute
- einwertige bzw. mehrwertige Attribute
- abgeleitete und gespeicherte Attribute
- Nullwert

Einfache Attribute sind atomar, das heißt, das Attribut ist nicht in weitere Bestandteile zerlegbar, während zusammengesetzte Attribute aus mindestens zwei hierarchisch geordneten Attributen zusammensetzt werden. In Abbildung 5-4 wird diese Problematik veranschaulicht.

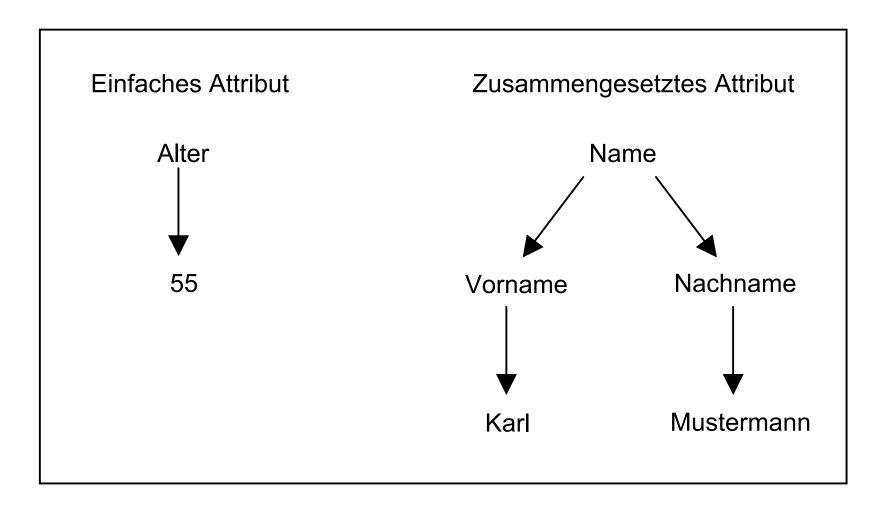


Abbildung 5-4: einfaches und zusammengesetztes Attribut

Das einwertige Attribut beinhaltet genau einen einzigen Wert für eine bestimmte Entität zu einem bestimmten Zeitpunkt. Einwertige Attribute sind die am häufigsten vorkommenden Attribute beim Datenbankentwurf.
Mehrwertige Attribute besitzen im Gegensatz zu den einwertigen Attributen mehrere Werte für eine bestimmte Entität zu einem bestimmten Zeitpunkt. Dabei kann durch eine Unter- und Obergrenze angegeben werden, wie niedrig beziehungsweise hoch die Anzahl der Werte für eine Entität sein darf. Diese Einteilung in Unter- und Obergrenzen wird auch die (min, max) -Notation genannt. Einwertige und mehrwertige Attribute werden in Abbildung 5-5 erläutert.

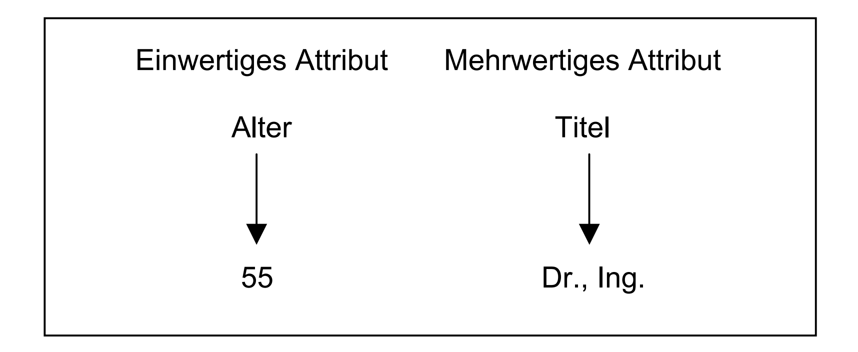


Abbildung 5-5: einwertiges und mehrwertiges Attribut

Abgeleitete Attribute stehen zu einem oder mehreren gespeicherten Attributen in Beziehung. Folglich kann der Wert eines abgeleiteten Attributes durch ein oder mehrere gespeicherte Attribute bestimmt werden. Beispielsweise kann durch eine bestimmte Anzahl gekaufter Lizenzen und deren Einzelpreis als gespeicherte Attribute der Gesamtpreis als abgeleitetes Attribut ermittelt werden.

Es kann vorkommen, dass für ein bestimmtes Attribut einer Entität kein zutreffender Wert vorhanden oder kein Wert bekannt ist. In solch einem Fall gibt es die Möglichkeit, diesem Attribut einen Nullwert zuzuweisen. Der Nullwert kann sozusagen als Füllwert benutzt werden.

5.3. Anwendung des Entity-Relationship-Modells

5.3.1. Einleitung

In diesem Kapitel werden die bereitgestellten Informationen über das License-Management-System in ein Entity-Relationship-Modell umgesetzt. Dazu werden die in Kapitel 4. „Anforderungsanalyse an das License-Management-System“ erarbeiteten Informationen herangezogen, um die entsprechenden Entitäten, deren Attribute sowie die zwischen den Entitäten bestehenden Beziehungen zu analysieren.
Dazu werden im folgenden Kapitel die Entitäten, Beziehungen und deren Attribute verfeinert, bevor abschließend ein konsistentes Entity-Relationship-Modell dargestellt wird, das für den darauf aufbauenden logischen Datenbankentwurf notwendig ist.

5.3.2. Umsetzung der Ergebnisse der Informationsanforderung

In den in Kapitel 4. aufgeführten Informationsanforderungen wurden die durch Befragung der zukünftigen Anwender und Sachkundigen des License-Management-System gesammelten Informationen in schriftlicher Form dargestellt. Hierbei wurden die unterschiedlichen Anforderungen an den konzeptionellen Datenbankentwurf des License-Management-System sichtbar. Der Aufbau der textuellen Struktur gliedert sich dabei in die Beschreibung der Entitäten sowie die Auffindung ihrer Beziehungen und Attribute.

5.3.2.1. Entitäts- und Attributverfeinerung

Die in der Informationsanforderung ermittelten Kernbereiche des License-Management-System werden hier in Form von Entitätstypen und deren Attribute dargestellt und beschrieben.

<u>Kernbereich Vertrag</u>

Der erste ermittelte Kernbereich ist der Vertrag. Diesem Kernbereich sind Entitätstypen zugeordnet, welche diesen näher beschreiben und logisch zusammenhängen.
Der Entitätstyp „VERTRAG“ verkörpert eine Abstraktion gleichartiger Entitäten, was in diesem Fall die unterschiedlichen Verträge sind. Eine Vertragsausprägung kann dabei eine Software-Lizenz, eine Wartungs-Lizenz oder eine gekaufte Vollversion sein.
Dem Entitätstyp „VERTRAG“ werden logisch zusammenhängende Attribute zugewiesen. Diese beschreibenden Attribute stellen konkrete Ausprägungen dar und sind für die Speicherung von spezifischen Werten notwendig.
Der Entitätstyp „VERTRAG“ beinhaltet das Attribut Vertragsnummer (vertrag_nr), welches einen Datensatz eindeutig aus dem aktiven Datenbestand identifiziert. Des Weiteren besitzt eine Vertragsentität eine festgelegte individuelle Vertragslaufzeit (laufzeit). Dieses Attribut informiert den Anwender über den Nutzungszeitraum des vorhandenen Produkts. Das Vertragsvolumen wird durch das Attribut (anzahl) festgelegt und kann unter Umständen den Einzelpreis eines Produkts bestimmen, welcher sich an den Rabattkonditionen des Herstellers ausrichtet.
Um eine Vertragsentität eindeutig aus dem Entitätstyp „VERTRAG“ zu identifizieren, ist es notwendig, diesem einen Primärschlüssel, <pi> zuzuordnen, wie in Kap. 5.2.1. „Entität, Entitätsmenge, Entitätstyp und Schlüssel“ erläutert. Das Attribut „vertrag_nr“ eignet sich für die Identifikation eines Vertrags aus dem Datenbestand des License-Management-System, da dieses den Minimalitätsanforderungen eines Primärschlüssels entspricht und eindeutig identifizierbar ist.

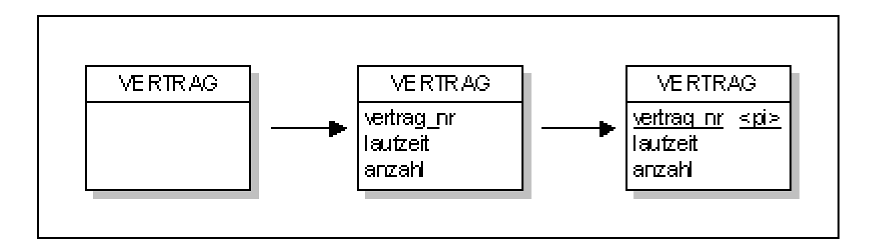


Abbildung 5-6: Verfeinerung des Entitätstyps „VERTRAG“

Um die Entität „VERTRAG“ ausreichend beschreiben zu können, wird der Entitätstyp „VERTRAGSART“ gebildet. Dies ist notwendig, da Verträge dem Anwender in verschiedenen Ausprägungen wie zum Beispiel End-User-License oder Maintenance-License vorliegen können. Um diese Namen nicht wiederholt eingeben zu müssen, bekommt die Entität „VERTRAGSART“ den eindeutig identifizierenden Primärschlüssel (vertrag_nr). Durch dieses Attribut kann der Anwender auf die wiederholte Eingabe der gleichen Vertragsart verzichten und muss im Folgenden lediglich die als Stammdaten abgespeicherten Vertragsnummern eingeben. Der (vertragsname) wird als weiteres Attribut der Entität „VERTRAGSART“ verwendet, ebenso wie das Attribut (vertragsbeschreibung). Dieses Attribut beinhaltet die genaue Definition der Vertragsart und ermöglicht dem Anwender einen guten Überblick über die Konditionen der jeweiligen Vertragsart.

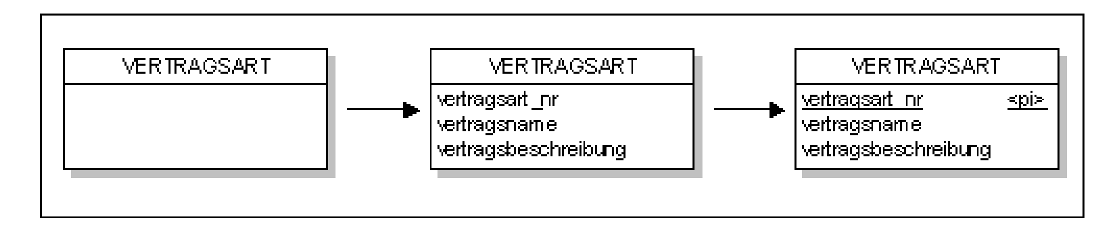


Abbildung 5-7: Verfeinerung des Entitätstyps „VERTRAGSART“

Neben der Entität „VERTRAGSART“ beschreibt auch die Entität „VERTRAGSKATEGORIE“ den Vertrag näher. In dieser Entität wird ein individueller Vertrag mithilfe des Attributs (vertragskategoriebezeichnung) unterschieden in Vollversion, Wartungs-Lizenz und Software-Lizenz. Um diese Bezeichnungen nicht wiederholt eingeben zu müssen und dem Anwender die Nutzung zu erleichtern, wird den einzelnen Vertragskategorien eindeutige Vertragskategorienummern zugeteilt. Diese bilden den Primärschlüssel dieser Entität, da sie die Vertragskategorie genau identifizieren. Anhand der Vertragskategorienummer ist für den späteren Anwender klar ersichtlich, um welche Art von Vertrag es sich handelt.



Abbildung 5-8: Verfeinerung des Entitätstyps „VERTRAGSKATEGORIE“

Um nachvollziehen zu können, wie ein Vertrag beschafft und eingekauft wird, muss die Entität „BESCHAFFUNG“ gebildet werden. Diese Entität ist notwendig, da alle Produkte eines Vertrags nicht zwangsläufig an einem Termin angeliefert oder Nachbestellungen zu einem Produkt getätigt werden. Durch das Attribut (beschaffungsdatum) wird der Beginn der Laufzeit des Produktes genau definiert und ist somit ein Primärschlüssel. Da dieses Attribut die Entität „BESCHAFFUNG“ aber nicht eindeutig identifizieren kann, muss ein weiteres Attribut eine Primärschlüsselfunktion übernehmen. Dazu wird das Attribut (sachnummer) herangezogen, da nur mithilfe dieses Attributs der zum Produkt zugehörige Vertrag eindeutig aufgefunden werden kann. Bei dieser Sachnummer handelt es sich um eine über das BMS vergebene QEV-Nummer oder um eine über BANF vergebene BANF-Nummer. Um den Beschaffungsweg auch für spätere Zeiten möglichst transparent zu halten, wird das Attribut (lieferschein_nr) als zusätzlich beschreibendes Attribut zum Entitätstyp „BESCHAFFUNG“ hinzugefügt. So ist immer nachvollziehbar, wer die Beschaffung angeliefert hat und wer sie entgegengenommen hat.



Abbildung 5-9. Verfeinerung des Entitätstyps „BESCHAFFUNG“

Der „LIEFERANT“ selbst bildet dabei einen eigenen Entitätstyp. Er wird eindeutig durch das Attribut (lieferant_nr) aus dem Datenbestand identifiziert. Die Vergabe des Primärschlüssels hält den Datenbestand konsistent und ermöglicht eine schnellere Nutzung der vorhandenen Daten. Attribute, die durch den Primärschlüssel eindeutig identifiziert werden, sind der (lieferantname) und (lieferant_kurz). Hier stehen im License-Management-System für den (lieferantname) die beiden Beschaffungswege Bestellantrag der neueren Form und Bestands-Management-System des Unternehmens DaimlerChrysler zur Auswahl. Das Attribut (lieferant_kurz) ist eine Kurzbezeichnung des Lieferantennamens und vereinigt die Bezeichnungen BANF und BMS unter sich. Diese Abkürzungen werden im täglichen Sprachgebrauch bei DaimlerChrysler und dadurch auch im License-Management-System verwendet.

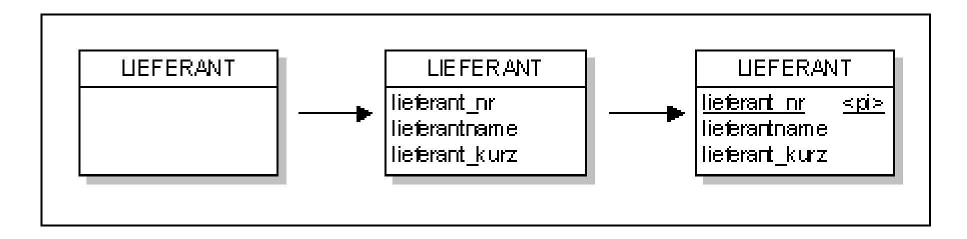


Abbildung 5-10: Verfeinerung des Entitätstyps „LIEFERANT“

Kernbereich Produkt

Aus den Informationsanforderungen an das License-Management-System geht hervor, dass die eingesetzte Software als Produkt bezeichnet wird. Dieser Name wird als Synonym im ER-Modell für die Bezeichnung einer Software weiterbenutzt und bildet den nächsten Kernbereich Produkt.

Um ein Produkt zu verwenden, ist es notwendig, dass für dieses ein entsprechender Vertrag existiert, um auf einem Computer installiert und dokumentiert werden zu können. Um ein Produkt zu erlangen, muss demnach ein Vertrag über eine bestimmte Anzahl an Lizenzen abgeschlossen werden, die das Nutzungsrecht für ein oder mehrere Produkte einräumen.

Der Entitätstyp „PRODUKT“ wird durch verschiedene Attribute beschrieben. Ein Produkt wird mit einer Produktnummer (produkt_nr) im Datenbestand aufgeführt. Diese (produkt_nr) wird als eindeutige Identifizierung der unterschiedlichen Produktausprägungen des Entitätstyps „PRODUKT“ benutzt und bildet den Primärschlüssel dieses Entitätstyps. Die (produkt_nr) ist einem (produktname) zugeordnet, welcher als beschreibendes Attribut im License-Management-System genutzt wird.



Abbildung 5-11: Verfeinerung des Entitätstyps „PRODUKT“

Das Produkt wird von einem Softwarehersteller entwickelt und durch eine Transaktion zur Verfügung gestellt. Der „HERSTELLER“ wird zum eigenen Entitätstyp und erhält den eindeutig identifizierenden Primärschlüssel (hersteller_nr). Dieser Primärschlüssel wird dem beschreibenden Attribut (herstellername) zugewiesen und sorgt durch die

einmalige Eingabe für die nötige Konsistenz der Daten, da im Folgenden auf die bereits eingegebenen Daten zugegriffen werden kann.



Abbildung 5-12: Verfeinerung des Entitätstyps „HERSTELLER“

Ein Produkt wird durch unterschiedliche Produktversionen unterschieden und kann für ein oder mehrere verschiedene Betriebssysteme und in unterschiedlichen Anwendungssprachen vorhanden sein.
Der Entitätstyp „VERSION“ wurde als eigenständiger Entitätstyp generiert, da zu einem Produkt unterschiedliche Versionen vorhanden sein können, aber die Trennung der beiden Entitätstypen eine mehrfache Eingabe der Produktdaten vermeidet. Die individuelle Produktversion wird durch eine spezifische (version_nr) identifiziert. Diese bildet den Primärschlüssel des Entitätstyps, der ein weiteres beschreibendes Attribut, die (bestandsmenge) zugewiesen bekommt. Die (bestandsmenge) gibt dem Anwender Aufschluss über die zur Ausgabe zur Verfügung stehenden Lizenzen der jeweiligen Produktversion.



Abbildung 5-13: Verfeinerung des Entitätstyps „VERSION“

Für Produktversionen stehen verschiedene Betriebssysteme zur Auswahl, welche in dem Entitätstyp „BETRIEBSSYSTEM“ definiert sind. Den Primärschlüssel dieses Entitätstyps bildet das Attribut (betriebssystem_nr). Dieses ist dem beschreibenden Attribut (betriebssystemname) zugewiesen und ermöglicht den Zugriff auf dessen Daten aus dem Datenbestand.

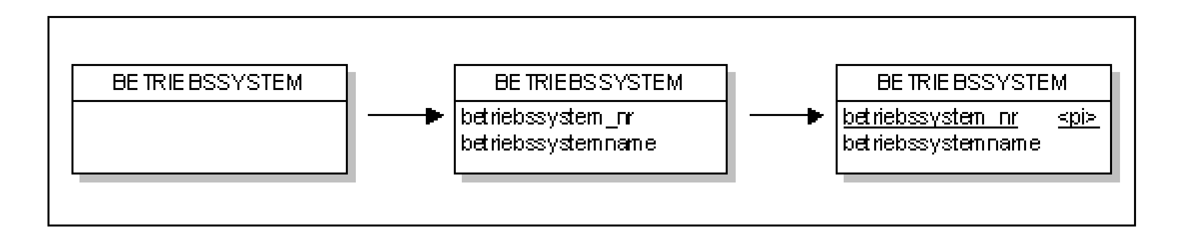


Abbildung 5-14: Verfeinerung des Entitätstyps „BETRIEBSSYSTEM“

Der Entitätstyp „SPRACHE“ wird gebildet, da für Produktversionen verschiedene Sprachen zur Verfügung stehen können. Den beschreibenden Attributen (sprachbezeichnung) und (sprache_kurz) wird der Primärschlüssel (sprache_nr) zugeteilt, der einen Zugriff auf diese Daten erleichtert. Durch das Attribut (sprache_kurz) sollen Eingabefehler und so eine Inkonsistenz der Daten verhindert werden, da aus einem vorgegebenen Feld von Sprachkurzbezeichnungen ausgewählt werden soll.

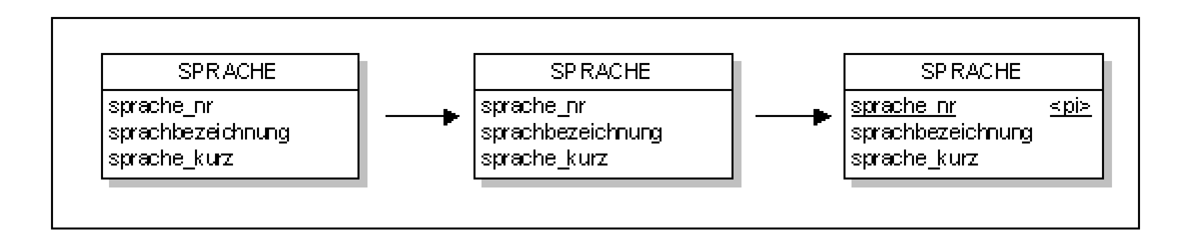


Abbildung 5-15: Verfeinerung des Entitätstyps „SPRACHE“

Um eine bestimmte Version eines Produkts nutzen zu können, kann die Freischaltung dieser durch einen Schlüssel erfolgen. Dieser „SCHLUESSEL“ bildet den nächsten Entitätstyp, welcher den Kernbereich „PRODUKT“ näher erläutert. Das Attribut (schluessel_nr) identifiziert Ausprägungen dieses Entitätstyps eindeutig und wird somit zum Primärschlüssel. Ihm werden weitere beschreibende Attribute zur Seite gestellt, die nähere Informationen zum Entitätstyp „SCHLUESSEL“ liefern. Das Attribut (schluessel) ist der zu nutzende Freischaltcode und wird der (schluessel_nr) zugewiesen. Das Attribut (flag_vergeben) wird hinzugefügt, um zu kennzeichnen, ob der jeweilige Schlüssel einer Produktversion bereits an einen Computer vergeben ist.

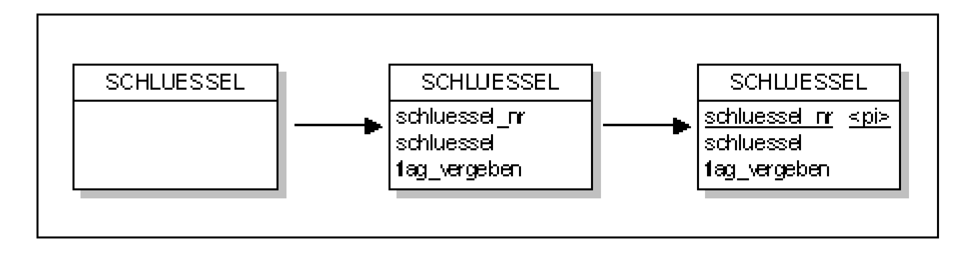


Abbildung 5-16: Verfeinerung des Entitätstyps „SCHLUESSEL“

Für den Entitätstyp „SCHLUESSEL“ stehen drei unterschiedliche Methoden der Schlüsselverwendung zur Verfügung. Je nach Vertragsvereinbarung zwischen dem Softwarehersteller und dem Unternehmen DaimlerChrysler stehen unterschiedliche Schlüsselmethoden zur Auswahl. Der Singlekey orientiert sich an der Anzahl der gekauften Lizenzen und besitzt für jede dieser Lizenzen einen individuellen Freischaltcode. Der Goldenkey bietet einen einzigen Freischaltcode für mehrere Lizenzen und der Concernkey wird verwendet, wenn kein Freischaltcode für ein lizenzpflichtiges Produkt benötigt wird. Diese verschiedenen Möglichkeiten werden in dem Entitätstyp „SCHLUESSEMETHODE“ abgebildet. Die genannten Methoden werden durch das beschreibende Attribut (schluesselmethodenname) definiert und durch den Primärschlüssel (schluesselmethode_nr) eindeutig aus dem Datenbestand identifizierbar. Durch die Wahl einer bestimmten Schlüsselmethode durch den Anwender wird der Entitätstyp „SCHLUESSEL“ direkt beeinflusst, da die Anzahl der Schlüssel durch die Schlüsselmethode definiert wird und der Anwender sieht, welche Schlüsselmethode für ein bestimmtes Produkt vorliegt.

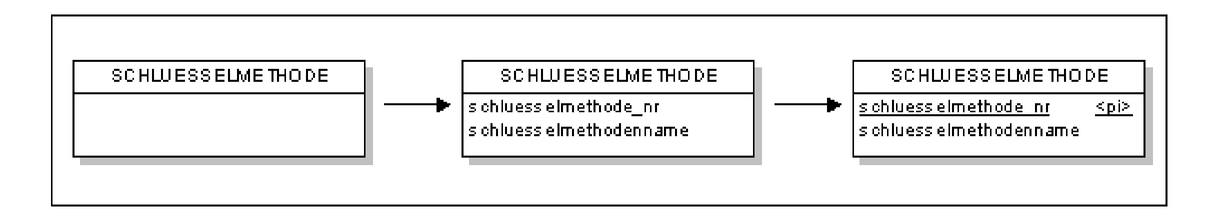


Abbildung 5-17: Verfeinerung des Entitätstyps „SCHLUESSELMETHODE“

Kernbereich Computer

Der Kernbereich Computer wurde in der Informationsanforderung als eindeutige Klassifikation eines realen Objekts identifiziert, das für die Speicherung von Produkten im License-Management-System erforderlich ist, siehe Kapitel 4.3.3. Kernbereich Computer.

Dabei wird der Entitätstyp durch charakteristische Eigenschaften beschrieben. Der Entitätstyp „COMPUTER“ besitzt, wie in Abbildung 5-18 dargestellt, einen Namen, der

den Computer bezeichnet. Dieser (computername) dient als Primärschlüssel und wird dem Computer bei der Beschaffung durch das Bestands-Management-System zugewiesen und unternehmensweit eindeutig identifizierbar. Zusätzlich bekommt der Entitätstyp das Attribut (inventar_nr) zugewiesen, welche durch das Werk Wörth vergeben wird und zur Identifikation eines Gegenstandes dient. Ein weiteres beschreibendes Attribut ist die (prozessoranzahl), die bei dem Abschluss einiger Lizenzverträge ausschlaggebend sein kann.



Abbildung 5-18: Verfeinerung des Entitätstyps „COMPUTER“

Der Entitätstyp „LVM_KLASSE“ gibt Auskunft über die Computerkategorie, wie zum Beispiel „PC-Workstation“ oder „PC-Notebook“. Sie beschreibt den Entitätstyp „COMPUTER“ näher und wird durch den Primärschlüssel „lvm_nr“ eindeutig identifiziert. Die genannten Computerkategorien werden dem Anwender durch das beschreibende Attribut (klassenbezeichnung) zur Verfügung gestellt.



Abbildung 5-19: Verfeinerung des Entitätstyps „LVM_KLASSE“

Der Archivpool

Neben den aufgeführten Kernbereichen steht der Archivpool. Dieser Bereich gilt nicht als Kernbereich, hat aber trotz allem eine systemerhaltende Funktion. Der Archivpool wird definiert durch eine Auswahl der beschreibenden Attribute der Kernbereiche. Mithilfe dieser Attribute wird es im Archiv möglich, Verträge, welche nicht mehr im aktiven Datenbestand sind, zu archivieren. Von dort können sie gegebenenfalls wieder reaktiviert und für weitere Vertragsverhandlungen genutzt werden. Dieser Entitätstyp

unterhält keine Beziehungen zu anderen Entitätstypen und wird als autonomer Entitätstyp im License-Management-System betrachtet.
Die in der folgenden Abbildung 5-20 aufgeführten Attribute ermöglichen das Speichern dieser Daten bis auf weiteres.

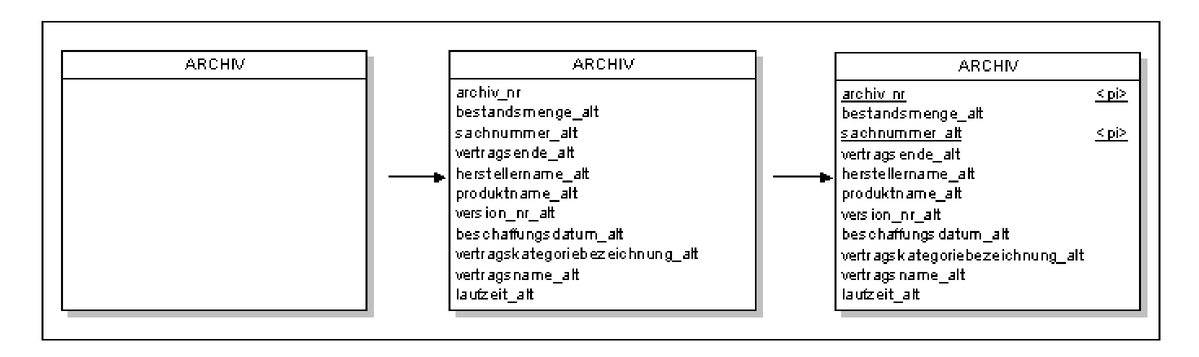


Abbildung 5-20: Verfeinerung des Entitätstyps „ARCHIV“

5.3.2.2. Beziehungsverfeinerung

Die Beziehungsverfeinerung im konzeptionellen Datenbankentwurf soll den Zusammenhang zwischen den identifizierten Entitätstypen aus „Entitäts- und Attributverfeinerung“ darstellen. Für ausführlichere Erklärungen sei hier auf Kapitel 5.2.2. „Beziehung, Beziehungsmenge und –typ, Rolle und Kardinalität“ verwiesen.
Die Beziehungen der Entitätstypen untereinander werden wie schon zuvor in die Kernbereiche Vertrag, Produkt und Computer unterteilt, bevor abschließend das Entity-Relationship-Modell mit all seinen Entitätstypen und Beziehungen dargestellt wird. Es wird dabei die Leseweise aus der Sicht beider Entitätstypen, die miteinander in Beziehung stehen, vorgestellt.

Beziehungen des Kernbereichs Vertrag

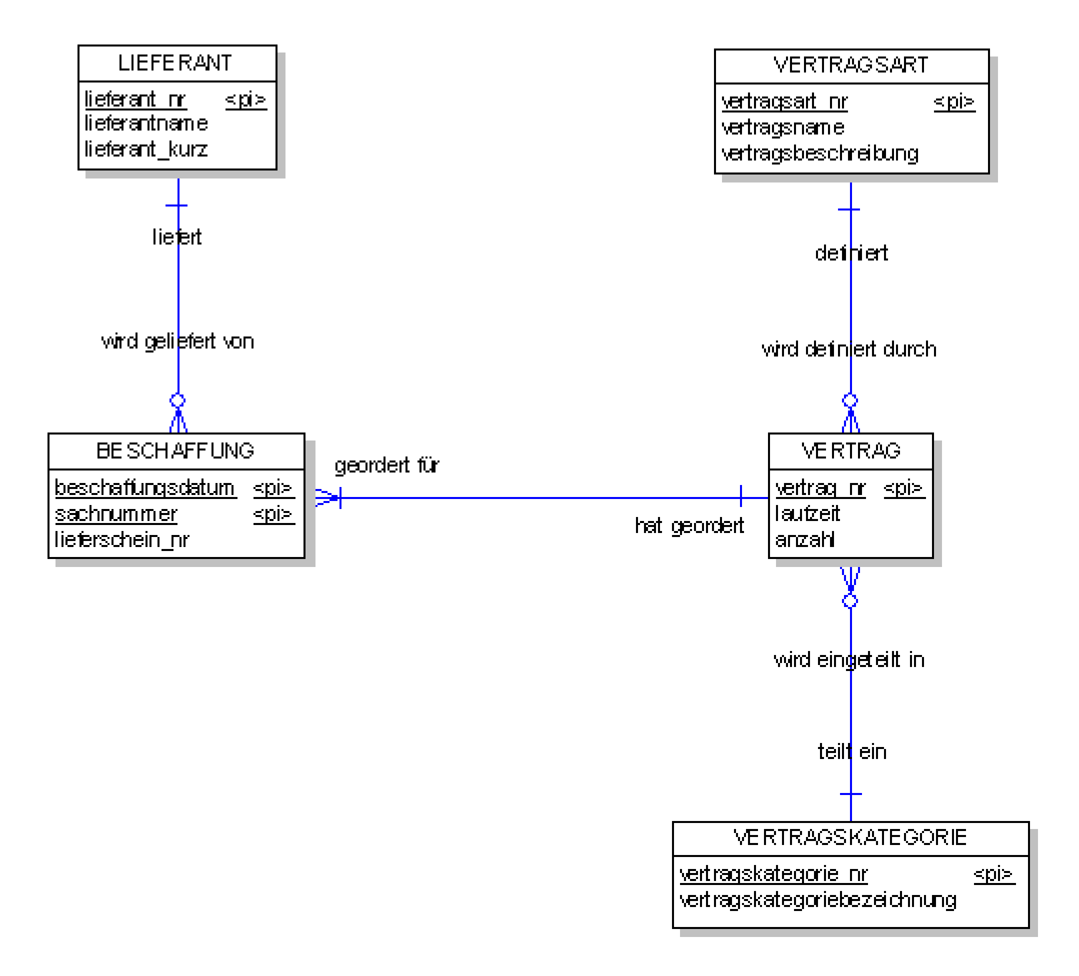


Abbildung 5-21: Beziehungen der Entitätstypen des Kernbereichs Vertrag

Der Kernbereich Vertrag umfasst mehrere logisch zusammengehörige Entitätstypen. Diese stehen miteinander in wechselseitiger Beziehung, welche in Abbildung 5-20 durch die Krähenfuß-Notation und die Rollenbezeichnungen dargestellt werden.

Die „One-to-many"-Beziehung zwischen den Entitätstypen „VERTRAG" und „VERTRAGSART" lautet:

Null, ein oder mehrere Verträge müssen definiert werden durch genau eine Vertragsart.

Genau eine Vertragsart kann null, ein oder mehrere Verträge definieren.

Die „One-to-many"-Beziehung zwischen den Entitätstypen „VERTRAG" und „VERTRAGSKATEGORIE" lautet:

Null, ein oder mehrere Verträge müssen eingeteilt werden in genau eine Vertragskategorie.

Genau eine Vertragskategorie kann null, ein oder mehrere Verträge einteilen.

Die „Many-to-one“-Beziehung zwischen den Entitätstypen „VERTRAG“ und „BESCHAFFUNG“ lautet:
Ein Vertrag muss ein oder mehrere Beschaffungen ordern.
Eine oder mehrere Beschaffungen können für genau einen Vertrag geordert werden.

Die „One-to-many“-Beziehung zwischen den Entitätstypen „BESCHAFFUNG“ und „LIEFERANT“ lautet:
Null, ein oder mehrere Beschaffungen müssen von genau einem Lieferanten geliefert werden.
Ein Lieferant kann null, eine oder mehrere Beschaffungen liefern.

Nach der Festlegung der Beziehungen zwischen den Entitätstypen wird deutlich, dass ein Vertrag durch die Vertragsart definiert und in Vertragskategorien eingeteilt wird. Dieser Vertrag muss in Form einer Beschaffung geordert werden und wird durch einen Lieferanten angeliefert, wie in Abbildung 5-21 ersichtlich wird.

Beziehungen des Kernbereichs Produkt

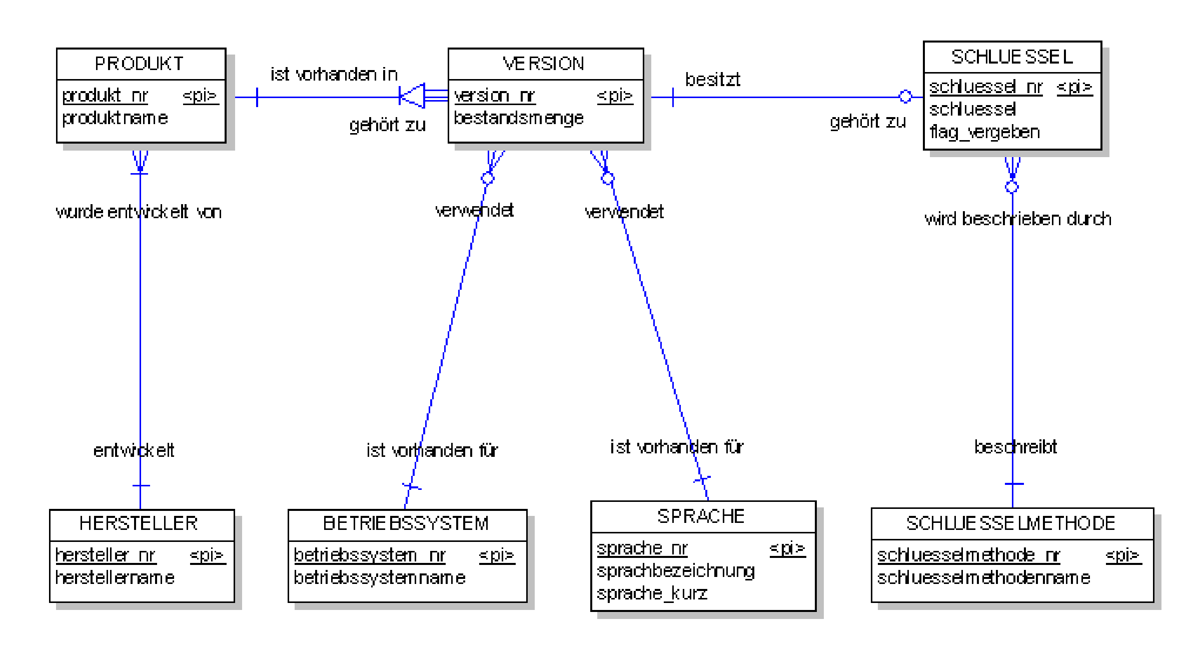


Abbildung 5-22: Beziehungen der Entitätstypen des Kernbereichs Produkt

Die „One-to-many“-Beziehung zwischen den Entitätstypen „PRODUKT“ und „HERSTELLER“ lautet:

Ein oder mehrere Produkte müssen von genau einem Hersteller entwickelt werden.
Ein Hersteller muss ein oder mehrere Produkte entwickeln.

Die „One-to-many”-Beziehung zwischen den Entitätstypen “PRODUKT” und “VERSION” lautet:
Ein Produkt muss vorhanden sein in einer oder mehreren Versionen.
Eine oder mehrere Versionen sind vorhanden für genau ein Produkt.
Dabei ist der Entitätstyp „VERSION“ eine schwache Entität und ist abhängig vom Entitätstyp „PRODUKT“. Eine Version ist nur eindeutig identifizierbar mit dem Primärschlüssel des Entitätstyps „PRODUKT“.

Die „One-to-many“-Beziehung zwischen den Entitätstypen „VERSION“ und „BETRIEBSSYSTEM“ lautet:
Null, eine oder mehrere Versionen können genau ein Betriebssystem verwenden.
Ein Betriebssystem muss für null, eine oder mehrere Versionen vorhanden sein.

Die „One-to-many“-Beziehung zwischen den Entitätstypen „VERSION“ und „SPRACHE“ lautet:
Null, eine oder mehrere Versionen können genau eine Sprache verwenden.
Eine Sprache muss für null, eine oder mehrere Versionen vorhanden sein.

Die „One-to-one“-Beziehung zwischen den Entitätstypen „VERSION“ und „SCHLUESSEL“ lautet:
Eine Version kann null oder einen Schlüssel besitzen.
Ein Schlüssel muss zu genau einer Version gehören.

Die „One-to-many“-Beziehung zwischen den Entitätstypen „SCHLUESSEL“ und „SCHLUESSELMETHODE“ lautet:
Null, ein oder mehrere Schlüssel müssen beschrieben werden durch genau eine Schlüsselmethode.
Eine Schlüsselmethode kann null, einen oder mehrere Schlüssel beschreiben.

Bei der Erklärung der Beziehungen des Kernbereichs Produkt wird deutlich, dass ein Produkt von einem Hersteller entwickelt wurde und in ein oder mehreren unterschiedlichen Versionen vorhanden ist. Diese Version wird verwendet in einem Betriebssystem und ist vorhanden in einer Sprache. Um eine Version ein zusetzten, kann es notwendig sein, diese mithilfe eines Schlüssels frei zuschalten. Der Schlüssel

selbst wird beschrieben durch eine Schlüsselmethode, welche die Schlüsselart näher definiert.

Beziehungen des Kernbereichs Computer

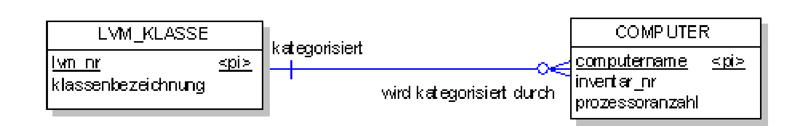


Abbildung 5-23: Beziehungen der Entitätstypen des Kernbereichs Computer

Die „One-to-many"-Beziehung zwischen den Entitätstypen „LVM_KLASSE" und „COMPUTER" lautet:
Eine LVM_Klasse kann null, ein oder mehrere Computer kategorisieren.
Null, ein oder mehrere Computer müssen durch genau eine LVM_Klasse kategorisiert werden.

Beziehungen zwischen den Kernbereichen Vertrag, Produkt und Computer

Bis zu dieser Stelle wurden die einzelnen Kernbereiche, die in der Informationsanforderung analysiert wurden, in Entitätstypen transformiert und ein logischer Zusammenhang zwischen diesen hergestellt. Dabei wurden die Entitätstypen mit ihren Attributen beschrieben und die Beziehungen der Entitätstypen untereinander erklärt und grafisch dargestellt.
Aus diesen Informationen resultiert das Entity-Relationship-Modell des License-Management-System, welches die gesammelten Informationen zu den einzelnen Kernbereichen und die Beziehungen dieser Kernbereiche untereinander darstellt. Das gesamte Modell, das im Anhang dargestellt ist, soll einen Gesamtüberblick gewährleisten und es dem Anwender ermöglichen, die gewonnenen Informationen anhand des Modells nachzuvollziehen. Das Modell selbst bietet die Grundlage für die Transformation in ein relationales Datenmodell.

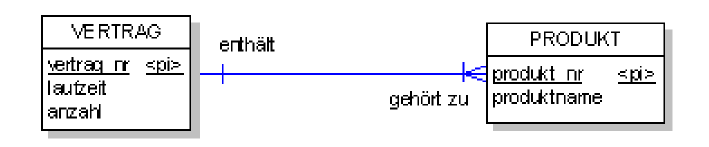


Abbildung 5-24: Schnittstelle zwischen den Kernbereichen Vertrag und Produkt

Die Beziehung zwischen den Kernbereichen Vertrag und Produkt ist eine „One-to-many“-Beziehung.
Ein Vertrag muss ein oder mehrere Produkte enthalten.
Ein oder mehrere Produkte müssen zu genau einem Vertrag gehören.

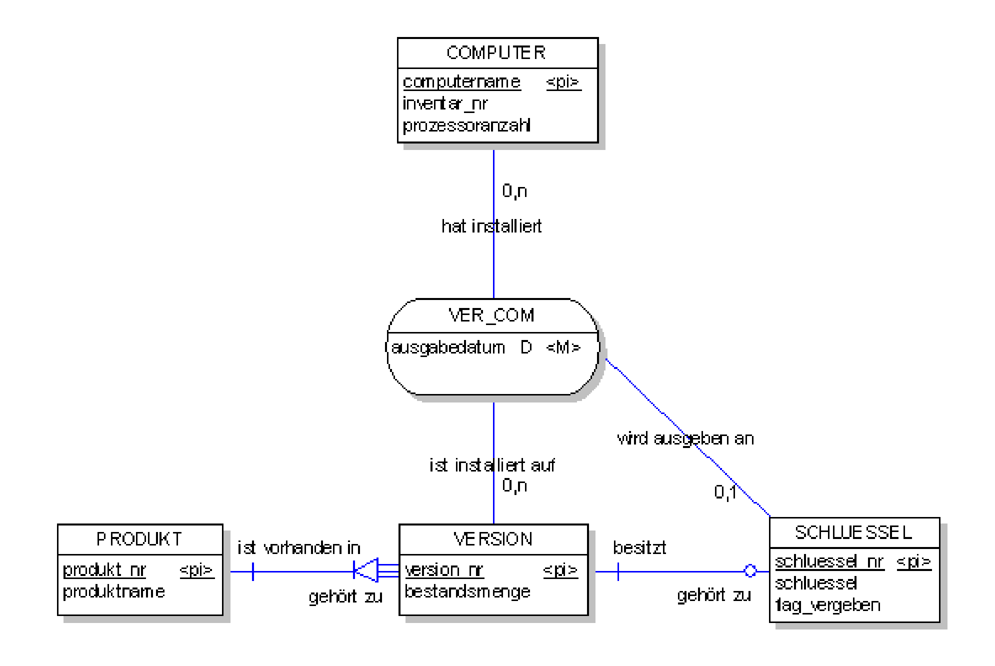


Abbildung 5-25: Schnittstelle zwischen den Kernbereichen Produkt und Computer

Die Beziehung zwischen den Kernbereichen Produkt und Computer wird zwischen den Entitätstypen „VERSION“ und „COMPUTER“ hergestellt.
Da eine Version einem Computer zugewiesen werden kann, ist es notwendig, zwischen den Relationen „VERSION“ und „COMPUTER“ eine Verbindung herzustellen. Da es sich hierbei um eine n:m-Beziehung handelt, wird hier eine neue Relation „VER_COM“ gebildet. Die neu entstandene Relation „VER_COM“ ist eine assoziative Entität, welche keinen eigenen Primärschlüssel besitzt. Sie bekommt die Primärschlüssel anderer Entitätstypen zugewiesen, wie im relationalen Modell ersichtlich wird. Die Beziehung zwischen den Entitätstypen „VERSION“ und „COMPUTER“ lautet:
Null, eine oder mehrere Versionen können auf null, ein oder mehreren Computern installiert sein.
Null, ein oder mehrere Computer können null, eine oder mehrere Versionen installiert haben.
Um diese Beziehung der Entitätstypen eindeutig unterscheiden zu können, ist es notwendig, diese in dem assoziativen Entitätstyp „VER_COM“ zu speichern. Mithilfe

des beim relationalen Modell zusammengesetzten Primärschlüssels können diese Tupel eindeutig identifiziert werden.

Die neu gebildete assoziative Entitätstyp „VER_COM“ besitzt eine Beziehung zum Entitätstyp „SCHLUESSEL“. Diese Beziehung ist notwendig, wenn ein Schlüssel mit der jeweiligen Produktversion an einen Computer ausgegeben wird. Die Beziehung lautet:
Null oder ein Schlüssel kann an null, ein oder mehrere Computer ausgegeben werden.

6. Der logische und implementierte Datenbankentwurf

6.1. Einleitung

Mithilfe des logischen Datenbankentwurfs ist es möglich, das konzeptionelle Datenbankschema in ein Zieldatenbankschema, welches in unserem Fall das relationale Datenbankschema ist, zu überführen. Dabei stellt der logische Datenbankentwurf ein Datenmodell dar, das in ein Datenbank-Management-System, wie in diesem Fall das relationale Datenbank-Management-System DB2 Version 8.1 von dem Unternehmen IBM, mithilfe der Data-Definition-Language (DDL) der relationalen Datenbanksprache SQL implementiert werden kann.
Eine detaillierte Erklärung des relationalen Modells sowie der genauen Umsetzung der Entitäten, Attribute und Beziehungen des ER-Modells in Relationen des relationalen Modells wird in Kapitel 6.2 „Grundlagen des relationalen Datenbankmodells" gegeben.
Im Kapitel 6.3 „Umsetzung des ER-Modells in das relationale Datenbankmodell" werden die gesammelten Grundlagen aus Kapitel 6.2 auf das ER-Modell der Lizenzverwaltung angewendet, um daraus das relationale Datenmodell zu erhalten.

6.2. Grundlagen des relationalen Datenmodells

Das relationale Datenmodell gehört zu den am weitesten verbreiteten logischen Datenbankmodellen und wurde von Edgar Codd von IBM Research in den frühen 70er Jahren entwickelt und veröffentlicht. Aufgrund seiner Einfachheit und seiner mathematischen Grundlagen, die auf der Theorie der Relationen und der Prädikatenlogik der 1. Ordnung basieren, fand das relationale Datenmodell umgehend Aufmerksamkeit in der Fachwelt. Das relationale Datenmodell von Codd stellt dem Anwender eine einfache Sicht auf die Daten in Form von Relationen zur Verfügung, die informell auch als zweidimensionalen Tabellen bezeichnet werden.
Für weitere Erklärungen werden hier die Begriffe „Tabelle" und „Relation" äquivalent benutzt.

6.2.1. Relation, Attribut, Domäne und Tupel

Die Daten eines relationalen Datenbankmodells werden in Tabellenform, in so genannten Relationen, gespeichert. Dabei besteht eine Tabelle aus einer festgelegten Anzahl an Spalten, welche die Attribute der Relation darstellen (intensionale Sicht) und einer variablen Anzahl von Zeilen (extensionale Sicht), welche auch als Tupel bezeichnet werden.

Formal bedeutet dies: Sind die Mengen D_1, D_2, ..., D_n, die nicht unbedingt disjunkt sein müssen, gegeben, dann wird R als Relation auf diese n Mengen bezeichnet, wenn sie eine Menge von n-Tupeln (d_1, d_2, ..., d_n) ist, sodass d_1 zu D_1, d_2 zu D_2, ... und d_n zu D_n gehört. Die Mengen D_i werden als Wertebereiche (engl. domain) von R und n wird als Grad (engl. degree) von R bezeichnet.

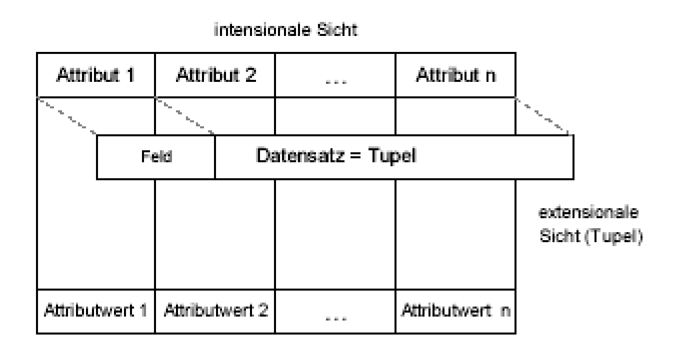


Abbildung 6-1: intensionale und extensionale Sicht auf eine Tabelle

Es bietet sich die Darstellung einer Relation als Tabelle an. Die Spalten der Tabelle werden als Attribute bezeichnet. Jedes Attribut hat einen eindeutigen Namen und wird immer durch den Namen, niemals durch seine Position, referenziert. Genauer gesagt bedeutet dies, dass Attribute keiner Ordnung unterliegen. Dabei hat jedes Attribut einen Wertebereich (engl. domain), welcher im relationalen Verständnis eng verwandt ist mit dem Konzept von Datentypen in Programmiersprachen. Der Wertebereich oder die Domäne eines Attributes gibt demnach an, welche Werte das Attribut annehmen kann. Wichtig ist dabei hervorzuheben, dass Attribute in einer Relation einzelne und nicht weiter zerlegbare Dateneinheiten sein müssen.

Jede Zeile einer Tabelle, auch Tupel genannt, ist eindeutig, das heißt, keine zwei Tupel dürfen in ihren Werten genau übereinstimmen. Diese Eigenschaft ergibt sich aus der Tatsache, dass eine Relation eine Menge von n Tupel ist. Eine Tupelmenge ist im relationalen Modell ungeordnet und frei von Duplikaten. Die Reihenfolge der Tupel ist folglich nicht genau definiert. Das ist ebenso bei den Attributen einer Relation der Fall, deren Reihenfolge nie genau definiert ist, was es mit sich bringt, dass nie das n-te

Attribut einer Relation angesprochen werden kann, sondern ein bestimmtes Attribut immer über seinen Namen angegeben werden muss.
Die Kardinalität einer Relation ist die Anzahl der Tupel in dieser Relation, die sie zu einem beliebigen Zeitpunkt enthält. Ist die Relation leer, kann die Kardinalität auch gleich null sein.

6.2.2. Kandidatenschlüssel, Primärschlüssel und Fremdschlüssel

Ein Attribut oder eine Menge von Attributen, deren Werte eindeutig eine Zeile einer Tabelle identifizieren, werden als Schlüsselkandidaten (engl. candidate key) bezeichnet. Beispielsweise kann ein Computer eindeutig durch seinen Computernamen oder durch seine Inventarnummer identifiziert werden.
Im Sinne des relationalen Datenbankdesigns ist es anschließend notwendig, aus den existierenden Schlüsselkandidaten einen geeigneten Primärschlüssel (engl. primary key) auszuwählen, der eine Zeile eindeutig identifiziert. Dieser Primärschlüssel wurde im Entity-Relationship-Modell als „primary identifier“ mit der Zeichenfolge <pi> herausgestellt. Neben dem Primärschlüssel werden den Tabellen Fremdschlüssel zugewiesen (engl. foreign key). Diese stellen die Beziehungen zwischen den Tabellen her. Der Fremdschlüssel wird in anderen Tabellen, zu denen eine Beziehung hergestellt wird, als zusätzliches Attribut aufgenommen.
Die Darstellung der oben genannten Schlüsseltypen ist mit Ausnahme des Kandidatenschlüssels in Abbildung 6-2 dargestellt. Der Primärschlüssel wird, wie im obigen Abschnitt beschrieben, unterstrichen mit schwarzer Schrift und einem nachfolgenden Schlüsselsymbol <pk> gekennzeichnet, während der Fremdschlüssel mit normaler Schrift und einem folgenden <fk> Symbol dargestellt wird.
Die Darstellung der relationalen Verbindungen verzichtet auf die im ER-Modell genutzte Krähenfuß-Notation und stellt die Referenz zwischen über- und untergeordneten Tabellen mithilfe einer Pfeilnotation dar. Diese Notation stellt die Beziehung von der untergeordneten Tabelle ausgehend zu der übergeordneten Tabelle dar. Der Pfeil zeigt somit auf die übergeordnete Tabelle, welche ihren Primärschlüssel der untergeordneten Tabelle als Fremdschlüssel zuweist.

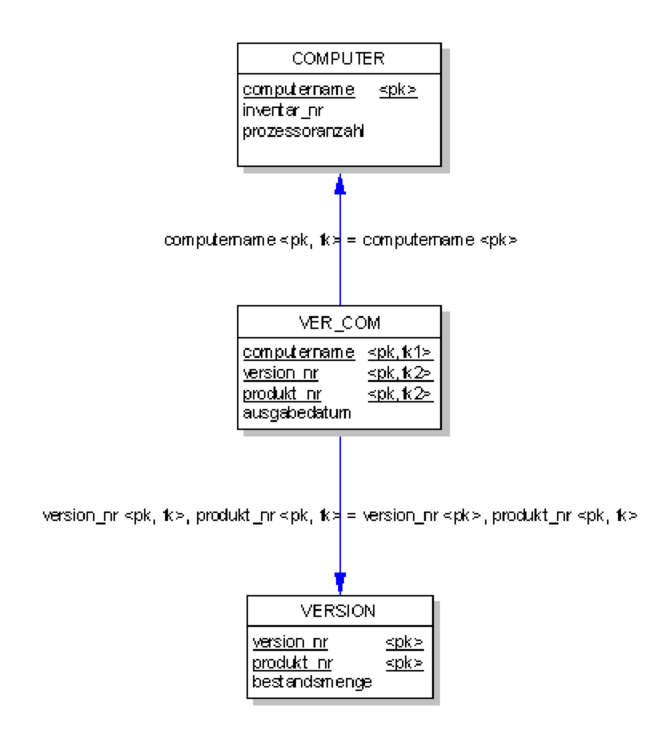


Abbildung 6-2: Primär- und Fremdschlüssel einer n:m-Beziehung

6.2.3. Entity- und referentielle Integrität

Die Entity-Integrität stellt im relationalen Datenbankentwurf sicher, dass jedes Tupel einen eindeutigen Schlüssel besitzt. Unter Eindeutigkeit wird verstanden, dass ein Schlüssel keinen Nullwert beinhalten darf. Die Definition für die Entity-Integrität lautet nach [sau98]: „Wenn ein Attribut die Komponente eines Primary-Key ist, dann darf dieses Attribut zu keinem Zeitpunkt einen NULL-Wert enthalten." Der Grund dafür ist, dass der Primärschlüssel benutzt wird, um einzelne Zeilen aus einer Tabelle zu identifizieren. Nullwerte für den Primärschlüssel würden implizieren, dass diejenigen Zeilen nicht aus dem Datenbestand identifiziert und unterschieden werden können.
Die Einhaltung der referentiellen Integrität wird zwischen zwei Relationen durch ihre Fremdschlüsselbeziehung definiert und dient der Einhaltung der Konsistenz zwischen den Tupel zweier Relationen.
Die referentielle Integrität implementiert referentielle Integritätsbedingungen im Datenbanksystem für Datenmanipulationsoperationen (Data Manipulation Language, DML), zu denen Einfügeoperationen (insert), Änderungsoperationen (update) und Entfernungsoperationen (delete) gehören.

Bei der Einfügeoperation INSERT ist es jederzeit möglich, eine Zeile in eine übergeordnete Tabelle einzufügen, ohne dass eine Aktion in den untergeordneten Tabellen ausgeführt wird. In einer abhängigen Tabelle können keine Zeilen eingefügt werden, wenn in der übergeordneten Tabelle kein entsprechender Schlüsselwert vorhanden ist.

Die Änderungsoperation UPDATE dient zur Aktualisierung der Tabellen. Zur Aktualisierung des Werts eines übergeordneten Schlüssels, der sich in einer übergeordneten Zeile befindet, muss man zunächst die Beziehung zu allen abhängigen Zeilen in den abhängigen Tabellen durch eine der folgenden Operationen beseitigen:

- Löschen der abhängigen Zeilen
- Aktualisierung des Fremdschlüssels in den abhängigen Tabellen, um sie danach mit anderen gültigen Schlüsselwerten zu versehen

Wird eine Zeile bei der Entfernungsoperation DELETE aus einer übergeordneten Tabelle gelöscht, wird geprüft, ob abhängige Zeilen in abhängigen Tabellen mit übereinstimmenden Fremdschlüsselwerten vorhanden sind. Werden abhängige Zeilen gefunden, sind verschiedene Aktionen möglich:

- RESTRICT verhindert, dass eine Zeile in der übergeordneten Tabelle gelöscht wird, wenn eine oder mehrere abhängige Zeilen gefunden werden.
- NO ACTION sichert das Vorhandensein einer übergeordneten Zeile für jede untergeordnete Zeile, nachdem die referentiellen Integritätsbedingungen angewendet wurden.
- CASCADE legt fest, dass durch das Löschen einer Zeile der übergeordneten Tabelle automatisch auch alle abhängigen Zeilen in der abhängigen Tabelle gelöscht werden.
- SET NULL bewirkt, dass durch das Löschen einer Zeile der übergeordneten Tabelle die Werte des Fremdschlüssels in allen abhängigen Zeilen auf NULL gesetzt werden.

6.2.4. Normalisierungstheorie

Bei der Normalisierung werden Daten unter Einhaltung vorgegebener Regeln aufgeteilt und verfeinert. Dabei soll das Aufteilen der Daten in der Art und Weise geschehen, dass sie am Ende den Normalisierungsregeln entsprechen. Normalisierungsgründe können dabei sein:

- Minimierung redundanter Daten
- Vermeidung von Änderungsanomalien
- Reduzierung inkonsistenter Daten
- Entwerfen von Datenstrukturen, die eine einfache Pflege erlauben

Für die ersten drei Normalformen ist die funktionale Abhängigkeit relevant. Um das Verständnis der Normalisierungstheorie zu erleichtern, sind die unten aufgeführten Definitionen von [sau98] hilfreich.

Definition: Funktionale Abhängigkeit
„In einer Relation R(A, B) ist das Attribut B von dem Attribut A funktional abhängig, falls zu jedem Wert des Attributs A genau ein Wert des Attributs B gehört."

Definition: Volle funktionale Abhängigkeit
„In einer Relation R(S1, S2, A) ist das Attribut A von den Attributen (Schlüsseln) S1, S2 voll funktional abhängig, wenn A von den zusammengesetzten Attributen (S1, S2) funktional abhängig ist, nicht aber von einem einzelnen Attribut S1 oder S2."

Definition: Transitive Abhängigkeit
„In einer Relation R(S, A, B) ist das Attribut B vom Attribut (Schlüssel) S (der auch ein zusammengesetzter Schlüssel sein kann) transitiv abhängig, wenn A von S funktional abhängig ist, S jedoch nicht von A, und B von A funktional abhängig ist."

Sind die funktionalen Abhängigkeiten geklärt, können die folgenden Normalisierungsregeln auf die Relationen des Datenmodells angewandt werden.
Es werden bei der Normalisierungstheorie sechs Normalisierungsregeln unterschieden:

1. Normalform
Eine Relation liegt in der ersten Normalform (1NF) vor, wenn jeder Attributwert eine atomare, nicht weiter zerlegbare Dateneinheit ist.

2. Normalform

Eine Relation liegt in der zweiten Normalform (2NF) vor, wenn sie in der ersten Normalform ist und jedes Nichtschlüsselattribut voll funktional vom Primärschlüssel abhängig ist, nicht aber von Schlüsselteilen.

3. Normalform

Eine Relation liegt in der dritten Normalform (3NF) vor, wenn sie in der zweiten Normalform ist und jedes Nichtschlüsselattribut nicht transitiv abhängig vom Primärschlüssel ist.

Boyce/Codd-Normalform

Eine Relation liegt in der Boyce/Codd-Normalform vor, wenn jede Determinante ein Candidate-Key ist.

4. Normalform

Eine Relation liegt in vierter Normalform (4NF) vor, wenn sie in der dritten Normalform ist und keine paarweise auftretenden mehrwertigen Abhängigkeiten enthält.

5. Normalform

Eine Relation liegt in der fünften Normalform (5NF) vor, wenn sie unter keinen Umständen aufgrund einer Verschmelzung einfacherer (d.h. weniger Attribute aufweisender) Relationen mit unterschiedlichen Schlüsseln rekonstruiert werden kann.

6.3. Umsetzung des ER-Modells in das relationale Datenmodell

Basierend auf den vorher genannten Grundlagen des relationalen Modells werden die Attribute der Entitätstypen mit den jeweiligen Wertebereichen versehen. Die Abbildungen der Entitätstypen entsprechen dabei relationalen Tabellen.

Die bestehenden Beziehungen zwischen den einzelnen Tabellen werden nach der Transformation der Entitätstypen in Relationen erläutert, die sowohl durch Fremdschlüssel als auch durch die Pfeilnotation der Modellierungssoftware PowerDesigner Evaluationsversion 10.1.0 des Unternehmens Sybase genauer definiert werden.

Anschließend werden die Phasen der Normalisierung zur Verfeinerung der Datenstruktur auf die Tabellen und ihre Beziehungen angewendet, um damit die Konsistenz und die referentielle Integrität der Daten zu überprüfen und sicher zustellen.

Um die so entstandenen Tabellen in ein Zieldatenbanksystem überführen zu können, welches in unserem Fall das Datenbank-Management-System DB2 Version 8.1 des Unternehmens IBM ist, müssen diese in SQL (Structured Query Language), genauer in der DDL von SQL, der Datendefinitionssprache, dargestellt werden. Das Datenbank-Management-System sowie dessen Konfiguration und Katalogisierung wurde von DaimlerChrysler im Werk Wörth zur Verfügung gestellt.

6.3.1. Umsetzung der Entitätstypen

Bei der Umsetzung der Entitätstypen des ER-Modells in Tabellen des relationalen Modells werden die Attribute des ER-Modells zu Attributen des relationalen Modells und die bereits vergebenen Primärschlüssel werden als identifizierendes Attribut weiterhin genutzt. Mithilfe des Primärschlüssels wird für die Einhaltung der Entity-Integrität des relationalen Datenmodells gesorgt, wie in Kapitel 6.2.3. „Entity- und referentielle Integrität“ erläutert. Des Weiteren müssen bei der Transformation des ER-Modells in das relationale Modell den Attributen die entsprechenden Wertebereiche zugewiesen werden. Die Domäne definiert diesen Wertebereich eines Attributs und stellt damit eine Art Datentyp dar. Diese Wertebereiche werden in der folgenden Abbildung aufgezeigt und die Umsetzung dieser Tabellen in die Datendefinitionssprache von SQL, wobei bei dieser Darstellung der Programmierung auf die Fremdschlüsselvergabe verzichtet wurde, da diese zu einem späteren Zeitpunkt ausführlich vorgestellt wird.
Die folgende Abbildung zeigt die Verbindung der Tabellen „COMPUTER“ und „VERSION“ mit ihren dazugehörigen Wertebereichen:

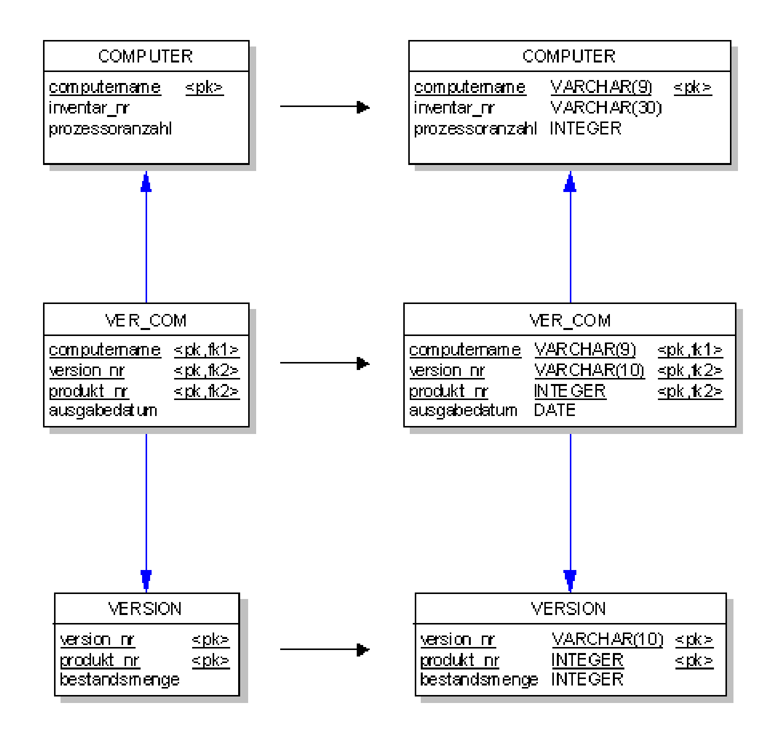


Abbildung 6-3: Wertebereiche der Relation „VERTRAG“, „VER_COM“, „VERSION“

Darstellung der Tabelle „COMPUTER“ in der Datendefinitionssprache von SQL:

```
CREATE TABLE COMPUTER (
    computername VARCHAR(9) NOT NULL,
    inventar_nr VARCHAR(30),
    prozessoranzahl INTEGER,
    PRIMARY KEY (computername)
);
```

Abbildung 6-4: Relationale Tabelle „VERTRAG“ in der DDL von SQL

Darstellung der Tabelle „VER_COM“ in der Datendefinitionssprache von SQL:

```
CREATE TABLE VER_COM (
    computername VARCHAR(9) NOT NULL,
    version_nr VARCHAR(10) NOT NULL,
    produkt_nr INTEGER NOT NULL,
    ausgabedatum DATE,
    PRIMARY KEY (computername, version_nr, produkt_nr)
);
```

Abbildung 6-5: Relationale Tabelle „VER_COM“ in der DDL von SQL

Darstellung der Tabelle „VERSION“ in der Datendefinitionssprache von SQL:

```
CREATE TABLE VERSION (
    version_nr VARCHAR(10) NOT NULL,
    produkt_nr INTEGER,
    bestandsmenge INTEGER,
    PRIMARY KEY (version_nr, produkt_nr)
 );
```

Abbildung 6-6: Relationale Tabelle „VERSION“ in der DDL von SQL

6.3.2. Umsetzung der referentiellen Integrität im relationalen Modell

Im Entity-Relationship-Modell wurden die logischen Zusammenhänge zwischen einzelnen Entitäten dargestellt. Es ist sinnvoll und nützlich, einige Daten in eigenen Entitäten zu definieren, wenn sich diese nur selten verändern, aber häufig in Gebrauch sind. Diese werden dann in eigenen Relationen genau definiert und mithilfe der referentiellen Integrität mit anderen Relationen in Beziehung gesetzt. Diese Verbindung zwischen den Relationen wird durch die Vergabe von Fremdschlüsseln erreicht. Durch die Bildung eigener Relationen für diese Daten wird verhindert, dass bei der Veränderung der Daten eine Inkonsistenz der Relation auftreten kann. Des Weiteren hat eine solche Ausgliederung auch den Vorteil der Genauigkeit und Eindeutigkeit der Daten, da es sich für den Anwender erübrigt, immer wieder die gleichen und langen Zeichenketten einzugeben, die zusätzlichen Speicherplatz kosten. Bei Stammdaten kann der Benutzer die gewünschten Daten aus einer Liste auswählen und muss diese nicht neu eingeben, was auch potenzielle Eingabefehler vermeidet. Diese könnten entstehen, wenn Daten fehlerhaft eingegeben werden und mit diesen Fehlern im aktiven Datenbestand gespeichert werden.

Im relationalen Modell des License-Management-System wird somit zwischen Bewegungsdaten und Stammdaten unterschieden. Bewegungsdaten unterliegen ständigen Veränderungen und müssen bei jeder Benutzung neu eingegeben werden. Stammdaten dagegen sind relativ konstante Daten, welche sich selten ändern und vom Benutzer nicht bei jeder Benutzung neu eingegeben werden müssen. Um diese beiden Datenkategorien leicht unterscheiden zu können, werden sie im relationalen Modell farblich unterschieden. Die Relationen der Bewegungsdaten werden gelb hinterlegt und die Relationen der Stammdaten mit blauem Hintergrund dargestellt.

Bei der Ausgliederung der Relationen werden künstliche Schlüssel mithilfe einer Sequenz generiert. Diese sollen nach [Zehn89] folgende Eigenschaften erfüllen:

- *Eindeutigkeit:*
  Jede Entität hat einen Identifikationsschlüsselwert, der anderweitig nie vorkommt. Der Schlüssel ist unveränderlich.
- *Laufende Zuteilbarkeit:*
  Eine neu auftretende Entität erhält ihren Identifikationsschlüssel sofort.
- *Kürze, Schreibbarkeit:*
  Ein Identifikationsschlüssel soll (relativ) kurz sein und leicht geschrieben werden können.

Eine Sequenz ist dabei eine spezielle Bereitstellung eines Datentyps in DB2 Version 8.1 von IBM. Dieser Datentyp ist eine intern automatisch generierte Zahl und dient hier zur Erzeugung von eindeutigen Attributwerten, welche als Primärschlüssel verwendet werden können.
Mithilfe der referentiellen Integrität werden die Beziehungen zwischen den einzelnen Relationen der Kernbereiche Vertrag, Produkt und Computer dargestellt. Der Aufbau dieser Beziehungen wird mithilfe der Fremdschlüsselvergabe erreicht, bei welcher die Primärschlüssel einer Relation einer untergeordneten Relation als Fremdschlüssel zugewiesen werden.

6.3.2.1. Beziehungen zwischen den Relationen des Kernbereichs Vertrag

LIEFERANT		
lieferant_nr	INTEGER	<pk>
lieferantname	VARCHAR(30)	
lieferant_kurz	VARCHAR(10)	

lieferant_nr <fk> = lieferant_nr <pk>

VERTRAGSART		
vertragsart_nr	INTEGER	<pk>
vertragsname	VARCHAR(30)	
vertragsbeschreibung	LONG VARCHAR	

vertragsart_nr <fk> = vertragsart_nr <pk>

BESCHAFFUNG		
beschaffungsdatum	DATE	<pk>
sachnummer	VARCHAR(20)	<pk>
lieferant_nr	INTEGER	<fk2>
vertrag_nr	INTEGER	<fk1>
lieferschein_nr	VARCHAR(40)	

vertrag_nr <fk> = vertrag_nr <pk>

VERTRAG		
vertrag_nr	INTEGER	<pk>
vertragsart_nr	INTEGER	<fk1>
vertragskategorie_nr	INTEGER	<fk2>
laufzeit	INTEGER	
anzahl	INTEGER	
einzelpreis	DECIMAL(8,2)	

vertragskategorie_nr <fk> = vertragskategorie_nr <pk>

VERTRAGSKATEGORIE		
vertragskategorie_nr	INTEGER	<pk>
vertragskategoriebezeichnung	VARCHAR(40)	

Abbildung 6-7: Beziehungen des Kernbereichs Vertrag

Diese Abbildung zeigt die Verbindung zwischen allen Relationen des Kernbereichs Vertrag. Diese Verbindungen werden im Folgenden einzeln erläutert.

<u>Beziehung zwischen den Relationen „LIEFERANT“ und „BESCHAFFUNG“:</u>

Jedes Produkt, welches im License-Management-System genutzt wird, muss beschafft und durch einen Vertrag definiert werden.

Die Relation „BESCHAFFUNG“ bekommt nicht nur einen Primärschlüssel, sondern dieser wird aus zwei Schlüsselteilen generiert. So werden das „beschaffungsdatum“ und die „sachnummer“ beide zu einem Primärschlüssel, da die Beschaffung nur durch diese beiden gemeinsam genau definiert werden kann und dem Benutzer entsprechende Informationen zur Verfügung stellt. Dies ist notwendig, da Beschaffungen zeitlich unabhängig voneinander getätigt werden können, wie zum Beispiel bei der Nachbestellung von weiteren Lizenzen zu einem Produkt. Als weiteres beschreibendes Attribut wird der Relation „BESCHAFFUNG“ die „lieferschein_nr“ hinzugefügt. Doch die Produkte müssen auch über einen bestimmten Beschaffungsweg geordert werden und so entsteht eine weitere Relation, welche im Datenmodell des License-Management-System als Relation „LIEFERANT“ bezeichnet

wird. Diese Relation kann unzweifelhaft durch seine „lieferant_nr“, die den Primärschlüssel darstellt, bestimmt werden. Weitere Attribute sind „lieferantname“ und „lieferant_kurz“. Der Primärschlüssel der übergeordneten Relation „LIEFERANT“ wird als Fremdschlüssel der Relation „BESCHAFFUNG“ zugeordnet.

Beziehung zwischen den Relationen „VERTRAG“ und „BESCHAFFUNG“

Der Primärschlüssel „vertrag_nr“ der Relation „VERTRAG“ wird als Fremdschlüssel der Relation „BESCHAFFUNG“ zugeteilt, die als eigenen zusammengesetzten Primärschlüssel das „beschaffungsdatum“ und die „sachnummer“ besitzt. Weitere beschreibende Attribute der Relation „VERTRAG“ sind „laufzeit“, „anzahl“ und „einzelpreis“. Durch die Zuordnung des Primärschlüssels des Vertrags zur Relation „BESCHAFFUNG“ wird zwischen diesen beiden Relationen eine Verbindung hergestellt.

Beziehung zwischen den Relationen „VERTRAGSART“ und „VERTRAG“

Die Relation „VERTRAG“ bekommt noch weitere Fremdschlüssel anderer Relationen zugewiesen. So hat die Relation „VERTRAGSART“ den Primärschlüssel „vertragsart_nr“ und die weiteren Attribute „vertragsname und „vertragsbeschreibung“. Durch diese Relation kann der Vertrag näher beschrieben werden und dem Anwender werden Daten in Form von Stammdaten zur Auswahl gestellt, aus denen er die gewünschte Vertragsart auswählen kann.

Die Zuordnung des Primärschlüssels „vertragsart_nr“ als Fremdschlüssel zu der Relation „VERTRAG“ stellt die Verbindung zwischen diesen beiden Relationen her.

Beziehung zwischen den Relationen „VERTRAGSKATEGORIE“ und „VERTRAG“

Auch die Vertragskategorie wird zur eigenen Relation „VERTRAGSKATEGORIE“ mit dem Primärschlüssel „vertragskategorie_nr“. Die Relation besitzt als weiteres beschreibendes Attribut besitzt die „vertragskategoriebezeichnung“. Auch dieser Primärschlüssel wird der Relation „VERTRAG“ als Fremdschlüssel zugeordnet.

6.3.2.2. Beziehungen zwischen den Relationen des Kernbereichs Produkt

SCHLUESSELMETHODE
schluesselmethode_nr INTEGER <pk>
schluesselmethodenname VARCHAR(20)

schluesselmethode_nr <fk> = schluesselmethode_nr <pk>

SCHLUESSEL
schluessel_nr INTEGER <pk>
version_nr VARCHAR(10) <fk1>
produkt_nr INTEGER <fk1>
schluesselmethode_nr INTEGER <fk2>
schluessel VARCHAR(30)
flag_vergeben VARCHAR(10)

version_nr <fk>, produkt_nr <fk> = version_nr <pk>, produkt_nr <pk, fk>

PRODUKT
produkt_nr INTEGER <pk>
hersteller_nr INTEGER <fk2>
vertrag_nr INTEGER <fk1>
produktname VARCHAR(30)

produkt_nr <pk> = produkt_nr <pk, fk>

VERSION
version_nr VARCHAR(10) <pk>
produkt_nr INTEGER <pk,fk1>
betriebssystem_nr INTEGER <fk2>
sprache_nr INTEGER <fk3>
bestandsmenge INTEGER

hersteller_nr <fk> = hersteller_nr <pk>

betriebssystem_nr <fk> = betriebssystem_nr <pk>

sprache_nr <fk> = sprache_nr <pk>

HERSTELLER
hersteller_nr INTEGER <pk>
herstellername VARCHAR(30)

BETRIEBSSYSTEM
betriebssystem_nr INTEGER <pk>
betriebssystemname VARCHAR(15)

SPRACHE
sprache_nr INTEGER <pk>
sprachbezeichnung VARCHAR(15)
sprache_kurz VARCHAR(5)

Abbildung 6-8: Beziehungen des Kernbereichs Produkt

Beziehung zwischen den Relationen „HERSTELLER“ und „PRODUKT“

Die Relation „HERSTELLER“ steht in Verbindung mit der Relation „PRODUKT“, indem diese Relation den Primärschlüssel „hersteller_nr“ als Fremdschlüssel zugewiesen bekommt. Die Relation „HERSTELLER“ hat als weiteres beschreibendes Attribut den „herstellername“.

Beziehung zwischen den Relationen “PRODUKT“ und „VERSION“

Die Relation „PRODUKT“ besitzt als Primärschlüssel das Attribut „produkt_nr“. Weitere Attribute sind bis zu dieser Stelle die „hersteller_nr“, welche als Fremdschlüssel der Relation „HERSTELLER“ zugewiesen wurde und das Attribut „produktname“. Diese Relation vergibt ihren Primärschlüssel als Fremdschlüssel an die Relation „VERSION“ und bildet dort gemeinsam mit dem Primärschlüsselteil „version_nr“ den gemeinsamen Primärschlüssel dieser Relation. Die Version ist nur eindeutig identifizierbar mit dem Primärschlüssel der Relation „PRODUKT“, da sie von dieser Relation abhängig ist. Ein weiteres beschreibendes Attribut ist die „bestandsmenge“, welche einen Überblick über die vorhandenen Lizenzen dieser Produktversion gibt.

Beziehung zwischen den Relationen „BETRIEBSSYSTEM“ und „VERSION“

Die Relation „BETRIEBSSYSTEM“ bildet eine eigenständige Relation, um falsche Eingaben zu vermeiden und eine einfache Handhabung der Daten in der späteren Anwendung zu gewährleisten.
Diese Relation besitzt die Attribute „betriebssystem_nr“ und „betriebssystemname“. Das Attribut „betriebssystem_nr“ wird dabei zum künstlichen Primärschlüssel dieser Relation, welcher durch eine Sequenz erzeugt wird.
Dieser Primärschlüssel wird wieder als Fremdschlüssel in die Relation „VERSION“ eingefügt und stellt auf diese Weise eine Beziehung zwischen den beiden her.

Beziehung zwischen den Relationen „SPRACHE“ und „VERSION“

Die Relation „SPRACHE“ bildet wie die Relation „BETRIEBSSYSTEM“ eine eigene Relation. Die Relation „SPRACHE“ erhält den Primärschlüssel „sprache_nr“ und durch die weiteren Attribute „sprachbezeichnung“ und „sprache_kurz“ wird es dem Anwender ermöglicht, die gewünschten Daten als Stammdaten zu behandeln. Dadurch wird neben Zeit auch Speicherplatz gespart und jede Art von Unstimmigkeiten durch beispielsweise unterschiedliche Abkürzungen der Sprachen wird verhindert. Der Primärschlüssel „sprache_nr“ wird als Fremdschlüssel der Relation „VERSION“ hinzugefügt, wodurch die Verbindung hergestellt wird.

Beziehung zwischen den Relationen „VERSION“ und „SCHLUESSEL“

Die Verbindung zur Relation „VERSION“ wird dadurch hergestellt, dass der zusammengesetzte Primärschlüssel der Relation „VERSION“, die „version_nr“ und die „produkt_nr“ als Fremdschlüssel in die Relation „SCHLUESSEL“ eingefügt wird. Der künstlich generierte Primärschlüssel der Relation „SCHLUESSEL“ ist die „schluessel_nr“, weitere beschreibende Attribute sind „schluessel“ und „flag_vergeben“.

Beziehung zwischen den Relationen „SCHLUESSELMETHODE“ und „SCHLUESSEL“

Neben den beschriebenen Attributen der Relation „SCHLUESSEL“ wird dieser durch die Verbindung mit der Relation „SCHLUESSELMETHODE“ noch ein weiterer Fremdschlüssel, die „schluesselmethode_nr“, hinzugefügt. Dieses Attribut ist der künstlich generierte Primärschlüssel der Relation „SCHLUESSELMETHODE“ und definiert diese Relation gemeinsam mit dem beschreibenden Attribut „schluesselmethodenname“.

6.3.2.3. Beziehung zwischen den Relationen des Kernbereichs Computer

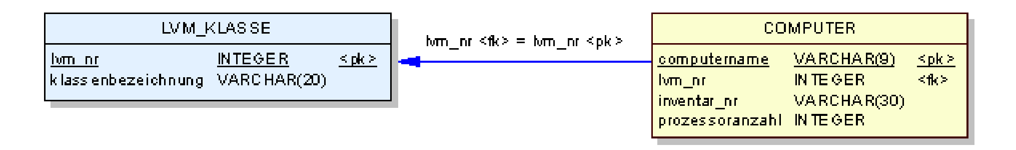


Abbildung 6-9: Beziehungen des Kernbereichs Computer

<u>Beziehung zwischen den Relationen „LVM_KLASSE“ und „COMPUTER“</u>
Die Relation „COMPUTER“ bekommt zusätzlich zu ihrem Primärschlüssel „computername“ einen Fremdschlüssel. Dieser wird von der Relation „LVM_KLASSE“ übergeben. Die LVM-Klasse kategorisiert die verwendeten Computer in unterschiedliche Gerätekategorien. Diese Relation hat neben dem beschreibenden Attribut „klassenbezeichnung“ die „lvm_nr“ zum künstlichen Primärschlüssel, der als Fremdschlüssel zur Relation „COMPUTER“ übergeht. Diese Relation hat als weitere beschreibende Attribute die „inventar_nr“, welche durch DaimlerChrysler Werk Wörth vergeben wird und die „prozessoranzahl“, welche bei der Vergabe einiger Lizenzen von Bedeutung ist.

6.3.2.4. Beziehung zwischen den Kernbereichen Vertrag und Produkt

SCHLUESSELMETHODE		
schluesselmethode_nr	INTEGER	<pk>
schluesselmethodenname	VARCHAR(20)	

schluesselmethode_nr <fk> = schluesselmethode_nr <pk>

SCHLUESSEL		
schluessel_nr	INTEGER	<pk>
version_nr	VARCHAR(10)	<fk1>
produkt_nr	INTEGER	<fk1>
schluesselmethode_nr	INTEGER	<fk2>
schluessel	VARCHAR(30)	
flag_vergeben	VARCHAR(10)	

HERSTELLER		
hersteller_nr	INTEGER	<pk>
herstellername	VARCHAR(30)	

hersteller_nr <fk> = hersteller_nr <pk>

version_nr <fk>, produkt_nr <fk> = version_nr <pk>, produkt_nr <pk, fk>

PRODUKT		
produkt_nr	INTEGER	<pk>
hersteller_nr	INTEGER	<fk2>
vertrag_nr	INTEGER	<fk1>
produktname	VARCHAR(30)	

produkt_nr <pk> = produkt_nr <pk, fk>

VERSION		
version_nr	VARCHAR(10)	<pk>
produkt_nr	INTEGER	<pk,fk1>
betriebssystem_nr	INTEGER	<fk2>
sprache_nr	INTEGER	<fk3>
bestandsmenge	INTEGER	

sprache_nr <fk> = sprache_nr <pk>

betriebssystem_nr <fk> = betriebssystem_nr <pk>

vertrag_nr <fk> = vertrag_nr <pk>

BETRIEBSSYSTEM		
betriebssystem_nr	INTEGER	<pk>
betriebssystemname	VARCHAR(15)	

SPRACHE		
sprache_nr	INTEGER	<pk>
sprachbezeichnung	VARCHAR(15)	
sprache_kurz	VARCHAR(5)	

VERTRAG		
vertrag_nr	INTEGER	<pk>
vertragsart_nr	INTEGER	<fk1>
vertragskategorie_nr	INTEGER	<fk2>
laufzeit	INTEGER	
anzahl	INTEGER	

vertrag_nr <fk> = vertrag_nr <pk>

BESCHAFFUNG		
beschaffungsdatum	DATE	<pk>
sachnummer	VARCHAR(20)	<pk>
lieferant_nr	INTEGER	<fk2>
vertrag_nr	INTEGER	<fk1>
lieferschein_nr	VARCHAR(40)	

vertragsart_nr <fk> = vertragsart_nr <pk>

vertragskategorie_nr <fk> = vertragskategorie_nr <pk>

lieferant_nr <fk> = lieferant_nr <pk>

VERTRAGSART		
vertragsart_nr	INTEGER	<pk>
vertragsname	VARCHAR(30)	
vertragsbeschreibung	LONG VARCHAR	

VERTRAGSKATEGORIE		
vertragskategorie_nr	INTEGER	<pk>
vertragskategoriebezeichnung	VARCHAR(40)	

LIEFERANT		
lieferant_nr	INTEGER	<pk>
lieferantname	VARCHAR(30)	
lieferant_kurz	VARCHAR(10)	

Abbildung 6-10: Beziehungen der Kernbereiche Vertrag und Produkt

Nachdem die einzelnen Relationen der Kernbereiche dargestellt wurden, werden Beziehungen zwischen diesen Kernbereichen hergestellt.

Die Relation „VERTRAG“ steht in Verbindung mit der Relation „PRODUKT“, da ein Vertrag immer über bestimmte Produkte abgeschlossen wird. So bekommen die Relation „PRODUKT“ den Primärschlüssel der Relation „VERTRAG“, die „vertrag_nr“, als Fremdschlüssel zugeteilt, wodurch eine Beziehung zwischen diesen beiden Relationen und so auch zwischen den Kernbereichen Produkt und Vertrag hergestellt wird.

6.3.2.5. Beziehung zwischen den Kernbereichen Produkt und Computer

SCHLUESSELMETHODE

schluesselmethode_nr	INTEGER	<pk>
schluesselmethodenname	VARCHAR(20)	

LVM_KLASSE

lvm_nr	INTEGER	<pk>
klassenbezeichnung	VARCHAR(20)	

lvm_nr <fk> = lvm_nr <pk>

COMPUTER

computername	VARCHAR(9)	<pk>
lvm_nr	INTEGER	<fk>
inventar_nr	VARCHAR(30)	
prozessoranzahl	INTEGER	

schluesselmethode_nr <fk> = schluesselmethode_nr <pk>

computername <pk, fk> = computername <pk>

SCHLUESSEL

schluessel_nr	INTEGER	<pk>
version_nr	VARCHAR(10)	<fk1>
produkt_nr	INTEGER	<fk1>
schluesselmethode_nr	INTEGER	<fk2>
schluessel	VARCHAR(30)	
flag_vergeben	VARCHAR(10)	

schluessel_nr <fk> = schluessel_nr <pk>

VER_COM

computername	VARCHAR(9)	<pk,fk1>
version_nr	VARCHAR(10)	<pk,fk2>
produkt_nr	INTEGER	<pk,fk2>
schluessel_nr	INTEGER	<fk3>
ausgabedatum	DATE	

version_nr <fk>, produkt_nr <fk> = version_nr <pk>, produkt_nr <pk, fk>

version_nr <pk, fk>, produkt_nr <pk, fk> = version_nr <pk>, produkt_nr <pk, fk>

PRODUKT

produkt_nr	INTEGER	<pk>
hersteller_nr	INTEGER	<fk2>
vertrag_nr	INTEGER	<fk1>
produktname	VARCHAR(30)	

produkt_nr <pk> = produkt_nr <pk, fk>

VERSION

version_nr	VARCHAR(10)	<pk>
produkt_nr	INTEGER	<pk,fk1>
betriebssystem_nr	INTEGER	<fk2>
sprache_nr	INTEGER	<fk3>
bestandsmenge	INTEGER	

sprache_nr <fk> = sprache_nr <pk>

hersteller_nr <fk> = hersteller_nr <pk>

betriebssystem_nr <fk> = betriebssystem_nr <pk>

HERSTELLER

hersteller_nr	INTEGER	<pk>
herstellername	VARCHAR(30)	

BETRIEBSSYSTEM

betriebssystem_nr	INTEGER	<pk>
betriebssystemname	VARCHAR(15)	

SPRACHE

sprache_nr	INTEGER	<pk>
sprachbezeichnung	VARCHAR(15)	
sprache_kurz	VARCHAR(5)	

Abbildung 6-11: Beziehungen der Kernbereiche Produkt und Computer

Bei der Betrachtung der möglichen Beziehungen zwischen den Kernbereichen Produkt und Computer stellt sich heraus, dass diese beiden Kernbereiche durch die Relationen „VERSION“ und „COMPUTER“ miteinander in Beziehung gesetzt werden müssen.

Diese ist notwendig, weil eine Version einem Computer zugewiesen werden kann und dadurch zwischen den Relationen „VERSION“ und „COMPUTER“ eine Verbindung entsteht. Da es sich hierbei um eine n:m-Beziehung handelt, wird hier eine neue Relation „VER_COM“ gebildet. Diese neu entstandene Relation bekommt die Primärschlüssel der Relationen „VERSION“ und „COMPUTER“ als Fremdschlüssel zugeordnet. Die Relation „VER_COM“ ist eine assoziative Entität und besitzt einen Primärschlüssel, welcher aus den Primärschlüsseln “version_nr“, „produkt_nr“ und „computername“ besteht. Ein weiteres Attribut dieser Relation ist „ausgabedatum“, welches das Datum der Versionsausgabe an einen Computer festhält.

Die Relation „VER_COM“ bekommt einen weiteren Primärschlüssel als Fremdschlüssel zugewiesen. Diese Fremdschlüsselvergabe entsteht durch die Verbindung zwischen dieser Relation mit der Relation „SCHLUESSEL“. Deren Primärschlüssel, die „schluessel_nr“, wird an die Relation „VER_COM“ als Fremdschlüssel vergeben. Diese Beziehung ist notwendig, wenn eine Produktversion mit einem Schlüssel an einen Computer vergeben wird. Dieser Schlüssel wird zurückgegeben, wenn dieser nicht mehr benötigt wird und steht für weitere Verwendungen zur Verfügung.

Nachdem die einzelnen Relationen der Kernbereiche dargestellt und erklärt wurden und die Kernbereiche untereinander verknüpft dargestellt sind, entsteht das vollständige relationale Datenmodell. Dieses Datenmodell zeigt die ursprünglichen Entitätstypen des Entity-Relationship-Modells in Form von Relationen, die Verbindungen zu anderen Relationen durch die referentielle Integrität in Form von Fremdschlüsselvergabe herstellen.
Das vollständige relationale Modell des License-Management-System ist im Anhang dieser Diplomarbeit abgebildet.

6.3.3. Normalisierung der Relationen

Die Darstellung der Relationen der Kernbereiche mit ihren Beziehungen untereinander wurden logisch zusammenhängend aufgebaut. Die Normalisierung soll im Folgenden überprüfen, ob diese Beziehungen konsistent sind und in eine Datenbank implementiert werden können.
Mögliche Inkonsistenzen und Redundanzen in den Tupel einer Relation ergeben sich, wenn Attribute nicht von dem Primärschlüssel anhängig sind. Wenn diese Daten durch Aktualisierungen, Einfüge- oder Löschoperationen verändert werden, wird die Relation inkonsistent und es können folgende Anomalien auftreten:

- Update-Anomalie
  Ändert sich beispielsweise die Produktnummer eines speziellen Produkts, dann muss in jedem Tupel der Relation die Produktnummer geändert werden, da diese Information mehrfach, also redundant, auftritt.
- Insert-Anomalie
  Wird ein Computer neu in den Datenbestand eingefügt, hat aber noch keine Produkte in Form von Produktversionen zugewiesen bekommen, ist dieser nicht über die Tabelle der genutzten Computer aufzufinden.

- Delete-Anomalie
  Wenn eine Version eines Produkts von einem Computer entfernt wird, steht diese dem weiteren Gebrauch nicht mehr zur Verfügung, obwohl das Nutzungsrecht weiterhin besteht.

Um diesen Anomalien vorzubeugen, wird die Normalisierung in den in Kapitel 6.2.4. „Normalisierungstheorie“ beschriebenen Schritten vorgenommen. Durch diese Normalisierungsschritte werden die Attribute der Relationen soweit normalisiert und verfeinert, dass alle Attribute, die nicht selbst einen Schlüssel darstellen, von dem Primärschlüssel abhängig sind. Dazu werden die Relationen des Kernbereichs Produkt als Attribute in eine einzige Relation gebracht, auf welche die Normalisierung beispielhaft angewandt wird.
Dabei ist es für unseren Fall ausreichend, das Datenmodell dieses Kernbereichs bis zur 3. Normalform zu normalisieren, da aus Gründen der Performance eine tiefere Modellierung nicht sinnvoll wäre.
Bevor jedoch die Normalisierung an diesem Beispiel vorgenommen werden kann, muss die funktionale Abhängigkeit (functional dependency) zwischen den Attributen näher betrachtet werden. Dabei wird in der folgenden Abbildung die funktionale Abhängigkeit der Attribute des Kernbereichs Produkt durch Pfeile dargestellt und die Bestandsteile der Relation Produkt, von denen andere Attribute funktional abhängig sind, werden unterstrichen hervorgehoben. Die Abhängigkeiten, welche im Folgenden einzeln vorgestellt und normalisiert werden, sind farblich unterschiedlich in funktionale Abhängigkeit gebracht.

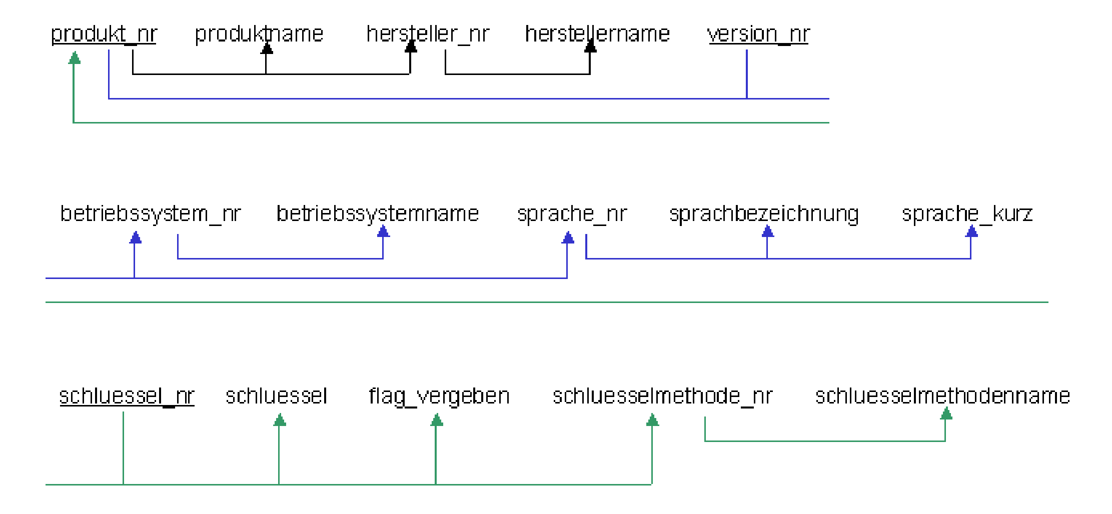


Abbildung 6-12: funktionale Abhängigkeiten der Relation „PRODUKT“

Hier sind alle Attribute des Kernbereichs Produkt mit ihren Abhängigkeiten aufgeschlüsselt und es wird klar ersichtlich, dass nicht alle Attribute eindeutig durch einen einzigen Primärschlüssel definierbar sind, sondern diverse Attribute selbst zu Primärschlüssel werden, um die Konsistenz der Daten zu wahren. Dies wird auch bei der folgenden Aufschlüsselung, die in Verbindung mit den Normalformen steht, deutlich. Abbildung 6-13 zeigt die funktionalen Abhängigkeiten zwischen den Attributen des Kernbereichs Produkt. Hier ist die erste Normalform erfüllt, da die Attribute der Relation nur atomare Werte beinhalten.

Erste ausgegliederte Relation:



Abbildung 6-13: „PRODUKT" in der 2. NF

Leseweise der 1. ausgegliederten Relation:

„Der „produktname" ist funktional abhängig von der „produkt_nr", die diesen Produktnamen bestimmt.
Die „hersteller_nr" ist funktional abhängig von der „produkt_nr", die diese Herstellernummer bestimmt.
Der „herstellername" ist funktional abhängig von der „hersteller_nr".

Die erste ausgegliederte Relation zeigt die funktionale Abhängigkeit zwischen den Attributen des Produkts und des Herstellers. Diese Relation befindet sich sowohl in der ersten als auch in der zweiten Normalform, da die Nichtschlüsselattribute von einem Primärschlüssel abhängen, welcher nicht zusammengesetzt ist. Die dritte Normalform ist an dieser Stelle nicht erfüllt.

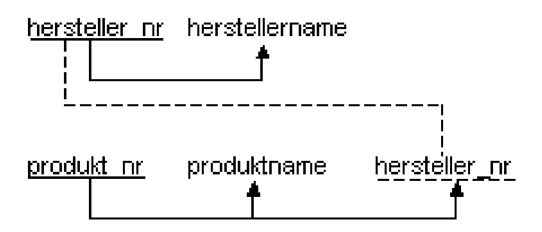


Abbildung 6-14: „PRODUKT" und „HERSTELLER" in der 3. NF

Hier ist die dritte Normalform erfüllt, da keine funktionale Abhängigkeit zwischen Attributen, die keine Schlüssel sind, besteht. Durch die Ausgliederung des Schlüssels „hersteller_nr“ mit seinem zugehörigen Attribut „herstellername“ ist keine transitive Abhängigkeit mehr vorhanden und die Verbindung zwischen den beiden Relationen wird durch die Fremdschlüsselvergabe hergestellt, welche hier durch die gestrichelte Linie dargestellt ist.

<u>Zweite ausgegliederte Relation:</u>

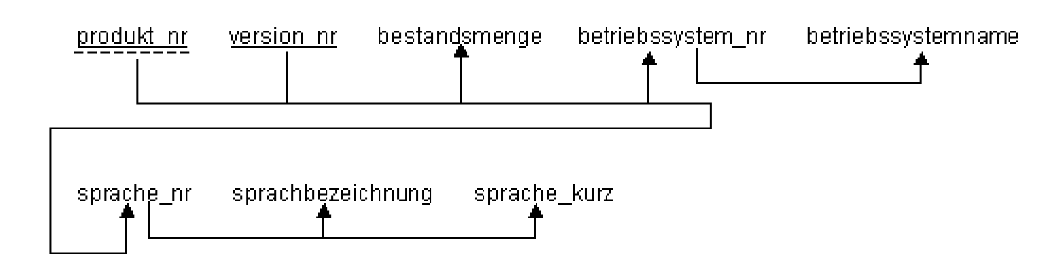


Abbildung 6-15: „VERSION“ in der 2. NF

Leseweise der 2. Relation:

„Die „bestandsmenge“, die „betriebssystem_nr“ und „sprache_nr“ sind funktional abhängig von dem aus „produkt_nr“ und „version_nr“ zusammengesetzten Primärschlüssel, welcher die beiden Attribute bestimmt. „Der „betriebssystemname“ ist funktional von der „betriebssystem_nr“ abhängig, welche diese bestimmt. Die „sprachbezeichnung“ und „sprache_kurz“ sind funktional abhängig von dem Primärschlüssel „sprache_nr“, welcher diese beiden Attribute bestimmt.“

Diese Relation befindet sich in der ersten und in der zweiten Normalform, da alle Attribute lediglich atomare Werte beinhalten und alle Nichtschlüsselattribute vom gesamten Primärschlüssel abhängig sind, nicht von Teilen dieses Primärschlüssels. Dies wird besonders bei dem Attribut „sprache_nr“ deutlich, welche von einem Primärschlüssel abhängt, welcher aus insgesamt zwei Schlüsselteilen besteht. Dabei ist die „sprache_nr“ von beiden Primärschlüsselteilen funktional abhängig, nicht nur von Teilen dieses Primärschlüssels.

3. Normalform:

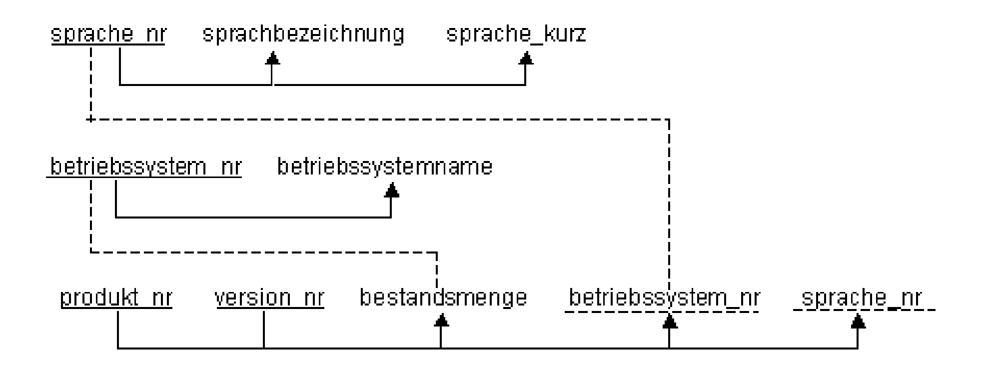


Abbildung 6-16: „VERSION", „BERTRIEBSSYSTEM" und „SPRACHE" in der 3. NF

Die dritte Normalform ist erfüllt, da keine transitive Abhängigkeit zwischen Attributen, die keine Schlüssel sind, besteht. Die Beziehung zwischen den Relationen wird durch die Fremdschlüsselvergabe hergestellt.

Dritte ausgegliederte Relation:

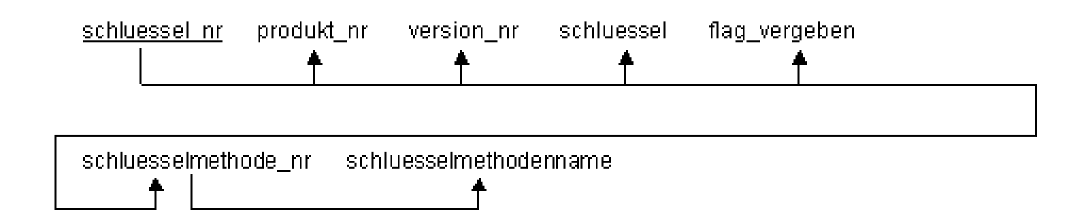


Abbildung 6-17: „SCHLUESSEL" in der 2. NF

Leseweise der 3. Relation:

„Die Attribute „produkt_nr", „version_nr", „schluessel", Flag_vergeben" und „schluesselmethode_nr" sind alle funktional vom Primärschlüssel „schluessel_nr" abhängig, welcher diese bestimmt. Der „schluesselmethodenname" ist funktional von der „schluesselmethode_nr" abhängig, welche dieses Attribut bestimmt."

Die dritte ausgegliederte Relation befindet sich in der ersten und in der zweiten Normalform, da alle Attribute dieser Relation nur atomare Werte beinhalten und jedes Nichtschlüsselattribut von den Primärschlüsseln abhängt, welche nicht zusammengesetzt sind.

3. Normalform:

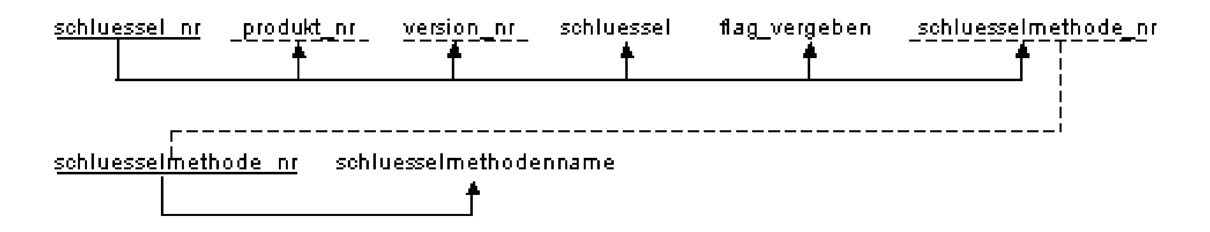


Abbildung 6-18: „SCHLUESSEL“ und „SCHLUESSELMETHODE“ in der 3. NF

Hier ist die dritte Normalform erfüllt, da keine funktionale Abhängigkeit zwischen Attributen, die keine Schlüssel sind, besteht. Die Beziehung „schluesselmethode_nr“ zu „schluesselmethodenname“ wurde ausgegliedert und in eine eigene Relation gebracht, damit keine transitive Abhängigkeit zwischen dem Nichtschlüsselattribut „schluesselmethodenname“ und den anderen Nichtschlüsselattributen besteht, welche vom Primärschlüssel „schluessel_nr“ abhängen. Die Beziehung zwischen der ausgegliederten Relation mit dem Primärschlüssel „schluesselmethode_nr“ und der Relation mit dem Primärschlüssel „schluessel_nr“ wird wiederum durch die Fremdschlüsselvergabe hergestellt, welche durch gestrichelte Linien verdeutlicht wird.

Die Umsetzung der Normalisierungsphasen bei der Relation „PRODUKT“ wurde nun exemplarisch durchgeführt und alle Beziehungen befinden sich in der 3. Normalform. Die Beziehung zwischen einzelnen ausgegliederten Relationen wurde durch die Fremdschlüsselvergabe hergestellt und ist im folgenden Bild durch die gestrichelten Verbindungslinien klar ersichtlich.

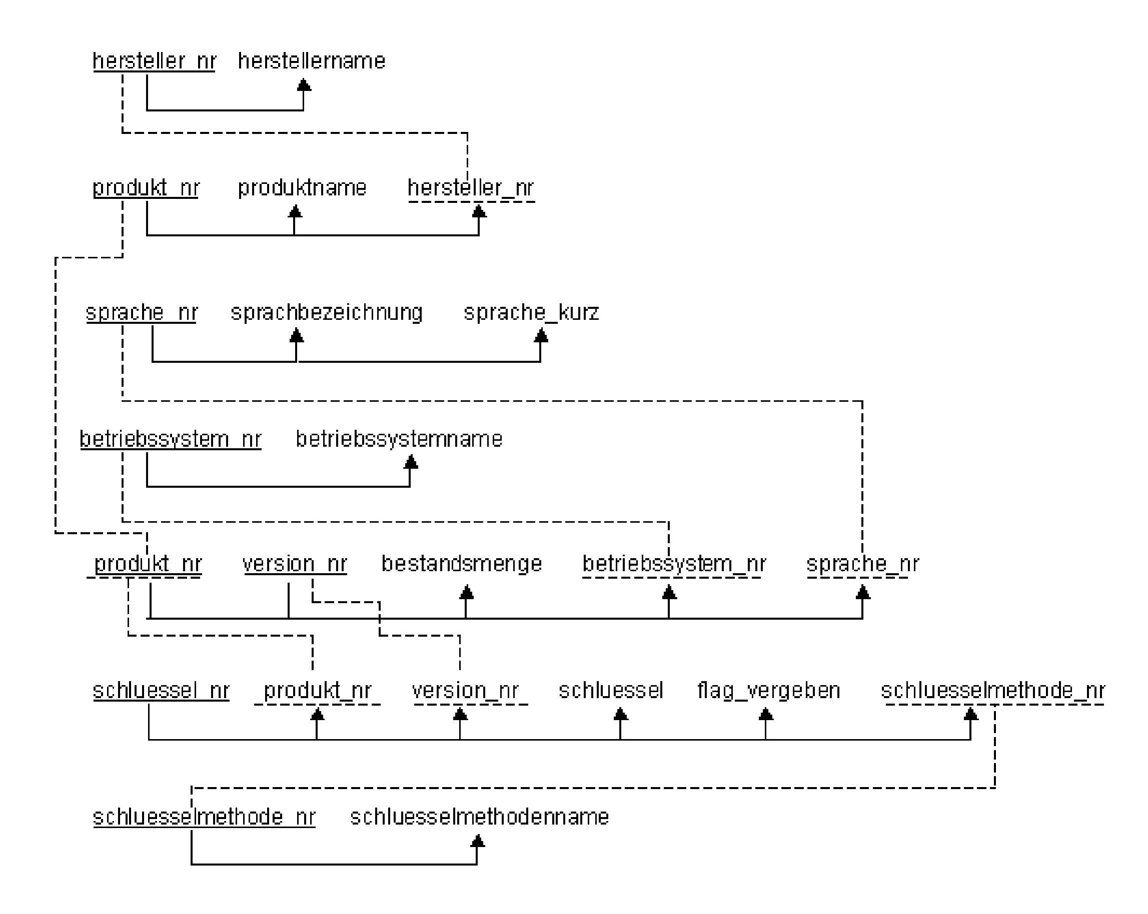


Abbildung 6-19: Fremdschlüsselbeziehung zwischen den normalisierten Relationen

6.4. Darstellung des implementierten relationalen Datenmodells

6.4.1. Einleitung

Das fertig gestellte relationale Datenmodell wird in das Datenbankmanagementsystem DB2 Version 8.1 des Unternehmens IBM implementiert. Die Implementierung erfolgt mithilfe der Structured Query Language (SQL), genauer gesagt, der Datendefinitionssprache (DDL) als Teil von SQL.
Die dargestellten verfeinerten Relationen werden mit den DDL-spezifischen Anweisungen in SQL umgesetzt und mit eindeutigen Primärschlüsseln versehen. Anschließend werden den Relationen die Fremdschlüssel zugewiesen, welche Beziehungen zu anderen Relationen, wie im obigen Modell durch die Fremdschlüsselvergabe dargestellt, herstellen. In besonderen Fällen wird für einen Primärschlüssel eine Sequenz benötigt, die einen eindeutigen Wert für den deklarierten Primärschlüssel generiert. Dies wird bei der künstlichen Generierung des

Primärschlüssels notwendig, da dieser keine eindeutig identifizierenden Attribute besitzt.

Die Vergabe der Fremdschlüssel bringt es mit sich, dass die referentiellen Integritätsbedingungen bezüglich Insert-, Update- und Delete-Operationen überprüft werden müssen. Hier muss angemerkt werden, dass nach den Entwicklungkonventionen von DaimlerChrysler AG Wörth die referentiellen Trigger-Funktionalitäten nicht in die Datenbank implementiert werden sollen, sondern diese vollständig in der Web-Applikation zu entwickeln sind.

Ist diese Überprüfung erfolgt und die referentielle Integrität für den aktiven Datenbestand des License-Management-System gewährleistet, erfolgt die Indizesvergabe für bestimmte Attribute. Diese beschleunigt die Verarbeitung bestimmter Datensätze und bringt so Performancevorteile mit sich.

Die Implementierung des relationalen Datenmodells in das Datenbank-Management-System wird in Form eines SQL-Skripts implementiert, welches im Anhang ausführlich beschrieben ist. An dieser Stelle wird die Implementierung in SQL beispielhaft an der Relation „PRODUKT“ aufgezeigt und erläutert.

6.4.2. Umsetzung der Relationen in SQL-Anweisungen

Mit den bisher gesammelten Informationen über das Datenmodell des License-Management-System werden nun Tabellen definiert, die diese Informationen in die DDL von SQL umsetzen. Die mit bestimmten Attributen versehene Relation „PRODUKT“ wird bei der Umsetzung in die Datendefinitionssprache von SQL wie folgt dargestellt:

```
CREATE TABLE PRODUKT (
    produkt_nr INTEGER NOT NULL,
    produktname VARCHAR(30),
    hersteller_nr INTEGER NOT NULL,
    vertrag_nr INTEGER,
    PRIMARY KEY (produkt_nr)
);
```

Abbildung 6-20: Relation „PRODUKT” in der DDL

Mithilfe des **CREATE TABLE**–Befehls wird eine leere Tabelle im Datenbank-Management-System erstellt und neu angelegt. Die bereits erläuterten Wertebereiche werden den Attributen dieser Tabelle zugewiesen. Mithilfe der **PRIMARY KEY**-Anweisung wird der Primärschlüssel als eindeutig identifizierendes Attribut dem Datenbank-Management-System bekannt gemacht.

6.4.3. Umsetzung des Fremdschlüssels in SQL-Anweisungen

Der Tabelle „PRODUKT" müssen nun Fremdschlüssel zugewiesen werden, um Verbindungen zu anderen Tabellen herzuleiten:

```
ALTER TABLE produkt
   ADD FOREIGN KEY (vertrag_nr)
   REFERENCES produkt (produkt_nr)
;
```

Abbildung 6-21: Fremdschlüsselvergabe der Relation „PRODUKT" in der DDL

Die Beziehung zwischen den Tabellen „VERTRAG" und „PRODUKT" wird dadurch hergestellt, dass die Tabelle „PRODUKT" den Primärschlüssel der Tabelle „VERTRAG", „vertrag_nr", als Fremdschlüssel zugeordnet bekommt. Die bereits bestehende Tabelle „PRODUKT" wird dabei durch die Anweisung **ALTER TABLE** verändert und bekommt durch die Anweisung **ADD FOREIGN KEY** den Primärschlüssel einer anderen Tabelle als Fremdschlüssel zugeteilt. Der Befehl **REFERENCES** referenziert die Tabelle „PRODUKT" mit dessen Primärschlüssel.

6.4.4. Generierung künstlicher Primärschlüssel in SQL

Da künstlich generierte Primärschlüssel keine eindeutig identifizierenden Attribute besitzen, muss diesen ein eindeutig identifizierender Wert zugewiesen werden. Die geschieht durch die Generierung von Sequenzen. Die folgende Abbildung zeigt die Definition einer Sequenz für das Datenbank-Management-System DB2:

```
CREATE SEQUENCE  seq_produkt
    START WITH 0
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    NO CYCLE
    CACHE 20
 ;
```

Abbildung 6-22: Sequenzgenerierung in der DDL

Mit dem Befehl **CREATE SEQUENCE** wird eine neue Sequenz für einen künstlich generierten Primärschlüssel erstellt. Der Zählerstand des Schlüssels fängt bei dem Wert 0 an, siehe Anweisung: **START WITH**. Wird ein neuer Datensatz einer Tabelle hinzugefügt, wird der Wert des Primärschlüssels durch die Anweisung **INCREMENT BY** um 1 erhöht. Die Anweisungen **NO MAXVALUE**/ **NO MINVALUE** bewirken, dass keine Grenze für einen maximalen beziehungsweise minimalen Wert erreicht wird. Da keine min-/maxvalue-Anweisungen definiert worden sind, ist es nicht notwendig, die Anweisung **NO CYCLE** auf **YES** zu setzen, da ein maximaler oder minimaler Wert nicht erreicht wird. Die letzte Anweisung **CACHE** besagt, dass die letzten 20 Werte im Speicher gehalten werden, um so einen schnelleren Zugang zu den individuellen Datensätzen zu gewährleisten. Diese Cache-Option erhöht dadurch die Performance der Datenbankanwendung.

6.4.5. Implementierung referentieller Integritätsregeln in SQL

„Integrität in Bezug auf eine Datenbank bedeutet, daß sie unversehrt ist, also keine widersprüchlichen Daten in ihr gespeichert sind.“[sau98] Die Vergabe von referenziellen Integritätsbedingungen ermöglicht, wie bereits beschrieben, dass der Datenbestand des License-Management-System konsistent bleibt und bei Insert-, Update- oder Delete-Operationen keine Inkonsistenzen auftreten. Im folgenden Ausschnitt des SQL-Scripts wird die referentielle Integritätsregel ON DELETE CASCADE dargestellt und näher beschrieben:

```
ALTER TABLE PRODUKT
    ADD FOREIGN KEY (vertrag_nr)
    REFERENCES produkt (produkt_nr)
    ON DELETE CASCADE
;
```

Abbildung 6-23: Integritätsbedingung der Relation „PRODUKT“ in der DDL

Der Zusatz **ON DELETE CASCADE** der **ALTER TABLE**-Anweisung bewirkt, dass die abhängigen Datensätze der Tabelle „PRODUKT“, dessen Fremdschlüssel dem Primärschlüssel der Tabelle „VERTRAG“ entsprechen, aus dem Datenbestand entfernt werden. Diese Operation auf die Daten nennt man kaskadierentes Löschen. Das heißt, wenn der Anwender ein Produkt aus dem Datenbestand entfernen möchte, wird auch gleich der dazugehörige Vertrag entfernt. Der Vorteil entsteht dadurch, dass man bei der späteren Anwendungsentwicklung keine verschachtelten Anweisungen über mehrere Tabellen benötigt, um einen Datensatz aus dem Datenbestand zu entfernen, sondern lediglich eine Delete-Operation auf eine Tabelle ausführt. Durch die Vergabe der referentiellen Integritätsbedingung **ON DELETE CASCADE** wird der entsprechende Datensatz, sowie die abhängigen Datensätze anderer Tabellen, welche durch Fremdschlüsselvergabe in Beziehung gesetzt wurden, entfernt.

6.4.6. Vergabe von Indizes in SQL

Die Verwendung von Indizes wird für die Primärschlüsselattribute vom Datenbank-Management-System DB2 automatisch durchgeführt. Um jedoch die Performance für Attribute, welche regelmäßig nach bestimmten Werten durchsucht werden, zu erhöhen, werden zusätzliche Indizes vergeben. Das folgende Beispiel zeigt die Vergabe eines Indizes für den Produktnamen, da dieser für die Suche nach bestimmten Produkten benötigt wird.

```
CREATE INDEX prod_bez_idx ON produkt (produktname DESC);
```

Abbildung 6-24: Indizes für das Attribut „produktname“ der Relation „PRODUKT“

Mit der Anweisung **`CREATE INDEX`** wird ein neuer Index erstellt. Dieser Index hat die Bezeichnung prod_bez_idx und teilt durch Angabe der Anweisung **`ON`** dem Datenbank-Management-System mit, dass dieser sich auf die Tabelle „PRODUKT“ bezieht. Die letzte Anweisung **`DESC`** besagt, dass sich der Index in absteigender Reihenfolge in Bezug auf das Attribut „produktname“ bildet.

6.4.7. Kontrolle des relationalen Datenmodells

Um die Zugriffsmöglichkeiten auf die relationalen Tabellen des License-Management-System zu testen, wird das Konzept der Datenbank-Views benutzt. Es werden Abfragen entwickelt um die Plausibilität für die im folgenden Kapitel entwickelte Web-Applikation gewährleisten zu können.

Eine View ist eine definierte Sichtweise auf Daten, die in der Datenbank gespeichert und vom Anwender abgefragt werden kann. Das bedeutet, eine View ist in der Lage, eine Abfrage über ein oder mehrere Tabellen mithilfe einer oder mehrerer Join-Operationen durchzuführen und dem Anwender die gesuchten Daten zur Verfügung zu stellen. Das Ergebnis einer View richtet sich dabei an den in der Anweisung definierten Abfragebedingungen aus. Wichtig ist hier zu bemerken, dass eine View keine physikalisch vorhandene Relation ist, sondern eine virtuelle Sichtweise auf einen Ausschnitt des relationalen Datenmodells. Diese Sichtweise wird als virtuelle Tabelle in Form einer View im Datenbank-Management-System gespeichert.

Im Folgenden wird in Abbildung 6-25 exemplarisch die View „v_installierteProdukte“ vorgestellt, welche unter anderen für den Test des relationalen Datenmodells benutzt wurde. Diese View ermittelt aus der Tabelle „COMPUTER“, „VER_COM“, „VERSION“, „PRODUKT“ und „HERSTELLER“ den Computernamen, den Herstellernamen, den Produktnamen, die entsprechende Versionsnummer sowie das Ausgabedatum des lizenzierten Softwareprodukts, auf welche die Suchbedingung in der WHERE-Klausel zutrifft. Die Filterung des gesuchten Daten aus dem Datenbestand wird dabei mithilfe der JOIN-Operatoren erreicht.

```
CREATE VIEW v_installierteProdukte AS
SELECT  c.computername,
      h.herstellername,
      p.produktname,
       vc.version_nr,
      vc.ausgabedatum
FROM computer c
RIGHT OUTER JOIN ver_com vc
RIGHT OUTER JOIN version vs
LEFT OUTER JOIN produkt p
LEFT OUTER JOIN hersteller h
ON h.hersteller_nr = p.hersteller_nr
ON vs.produkt_nr = p.produkt_nr
ON vc.version_nr = vs.version_nr
ON c.computername = vc.computername
WHERE c.computername LIKE 'qev10296'

;
```

Abbildung 6-25: View v_installierteProdukte in der DDL von SQL

7. Web-Applikation des License-Management-System

7.1. Einleitung

Dieses Kapitel stellt die konzeptionelle Umsetzung des relationalen Datenmodells vor. Es werden dabei Auszüge der Web-Applikation vorgestellt, die sowohl die gewünschten Funktionalitäten als auch die Anforderungen, die an diese Web-Applikation gestellt werden, beispielhaft beschreibt.
Dabei darf auch an dieser Stelle nicht vergessen werden, dass das Ziel dieser Diplomarbeit nicht ein fehlerfrei lauffähiges Programm ist, sondern ein Prototyp, der es dem Betrachter ermöglicht, einen Einblick in die Entwicklungsweise und in die Funktionalitäten der Benutzeroberfläche zu bekommen. Der Prototyp zeigt die komplexe Verknüpfung der unterschiedlichen Funktionen eines solchen Programms und ermöglicht einen Überblick über mögliche Realisierungen.
Die Realisierung der Web-Applikation wird unter Beachtung des MVC-Entwurfsmusters (Model-View-Controller) in das V4-Framework von DaimlerChrysler AG am Standort Wörth implementiert. Dieses Framework ermöglicht es dem Entwickler, Intranetanwendungen zu entwerfen und zur Verfügung zu stellen. Im Folgenden wird die prinzipielle Vorgehensweise der grafischen Benutzeroberflächenentwicklung aufgezeigt, um künftigen Entwicklern die Einarbeitung und Weiterentwicklung des License-Management-System zu erleichtern.

7.2. Einblick in das Model-View-Controller-Entwurfsmuster

Der Entwurf wieder verwendbarer objektorientierter Software ist schwer. Ein Entwurf muss den vorliegenden spezifischen Anforderungen genügen, jedoch auch allgemein genug sein, um zukünftigen Problemen und Anforderungen entgegen treten zu können. Man muss nicht jedes Problem von Grund auf neu angehen, sondern kann Lösungen verwenden, die zuvor erfolgreich eingesetzt wurden. In vielen objektorientierten Systemen lassen sich wiederkehrende Muster von Klassen und Objekten finden. Entwurfsmuster vereinfachen die Wiederverwendung von erfolgreichen Entwürfen und Architekturen. Ein Entwurfsmuster beschreibt ein häufig auftretendes Entwurfsproblem und präsentiert ein erprobtes Schema zu seiner Lösung.

Das hier verwendete MVC-Entwurfsmuster fand seine Gültigkeit bei der Entwicklung ereignisgesteuerter grafischer Benutzeroberflächen. Erstmals wurde dieses Konzept in der Programmiersprache Smalltalk-80 eingeführt. [Gam98] Mithilfe des MVC-Musters wird die Präsentationsschicht strukturiert. „Es entkoppelt die Darstellung (View) von der Datenhaltung (Model) und der Ablaufsteuerung (Controller) im System. Viele GUI-Frameworks basieren auf diesem Muster, beispielsweise die Java-Swing-Bibliothek."

Das MVC-Paradigma wird in drei Bereiche strukturiert:

Model

- stellt das Anwendungsobjekt dar
- enthält die Kernfunktionalität und die Daten

View

- stellt die Bildschirmrepräsentation dar
- Anwendungsobjekt erhält die Daten vom Modell

Controller

- Reaktion und Verarbeitung von Benutzereingaben
- Vermittler zwischen Model und View

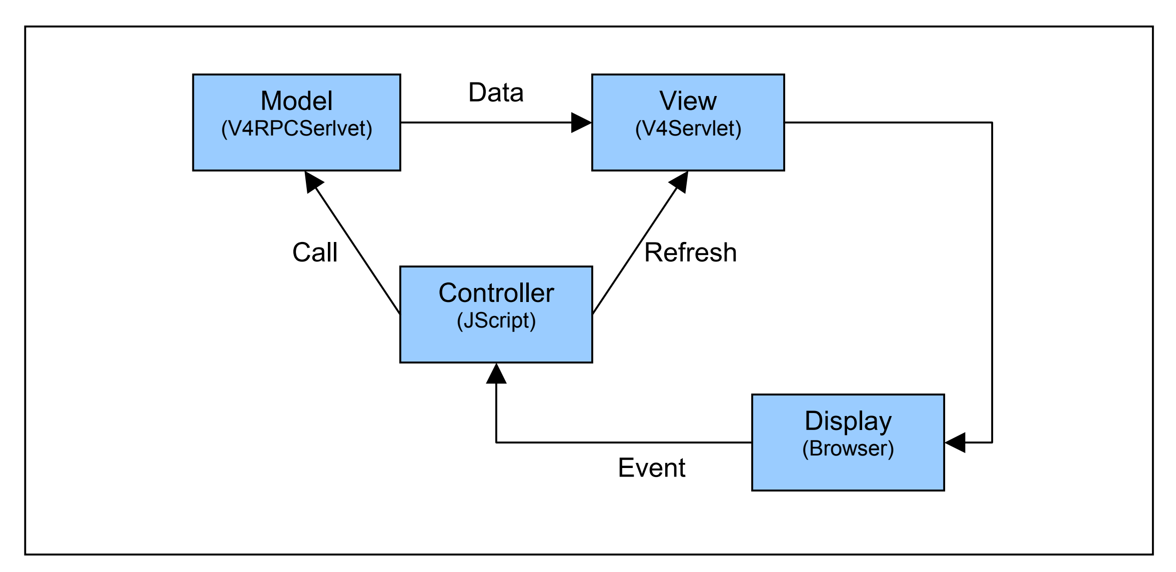


Abbildung 7-1: MVC-Interaktionsmuster

Das MVC-Muster benutzt dabei den Remote Procedure Call (RPC), der wie ein lokaler Prozedur- oder Methodenaufruf arbeitet. Dabei ist der Vorteil des RPC, dass nicht zwingend vorgegeben ist, dass sich die aufgerufene Komponente auf dem gleichen

Rechner befindet, es kann auch auf einem entfernten Rechnersystem sein. Es können also auch Prozeduren aufgerufen werden, die sich auf anderen Rechnern befinden. In diesem Fall wird auf den fremden Rechner zugegriffen. Das hier verwendete V4-Framework bedient sich des RPC-Mechanismus und stellt dem Anwender dadurch eine Vielzahl von Verbindungen zur Verfügung.
Die drei Bereiche des MVC-Musters, Model, View und Controller, werden bei der folgenden Web-Applikation an dem Beispiel „Vertrag suchen“ einzeln beschrieben und umgesetzt.

7.2.1. Entwicklung einer View am Beispiel “Vertrag suchen”

Die Entwicklung der grafischen Benutzeroberfläche wird im V4-Framework von DaimlerChrysler am Standort Wörth vorgenommen. Dabei werden V4Servlets, also serverseitige, dynamische Softwarekomponenten, genutzt, die es ermöglichen, die Funktionalität eines Servers zu erweitern. So können auf einzelne Anfrage genauere Ergebnisse erwartet werden, da das Servlet die weitergeleitete Anfrage (Request) auswertet und nur die dazugehörigen Antworten (Responses) zurückgibt. Servlets werden in Java programmiert und können als gewöhnliche Java-Klassen angesehen werden.
Das hier genutzte V4Servlet ist eine Erweiterung des Standart-HTTP-Servlets und bietet zusätzliche Funktionen für ein zusätzlich übergebenes User-Objekt. Das User-Objekt wird von den V4Servlets und dem im Model behandelten V4RPCServlet überprüft und identifiziert und autorisiert den jeweiligen Benutzer. Auf nähere Erläuterungen der Servlets und des User-Objekts kann an dieser Stelle verzichtet werden, da eine weitere Realisierung des Prototyps diese Informationen nicht erfordert.

Bei den V4Servlets ist es dem Entwickler möglich, aus einem vorgegebenen Pool an WebControlls auszuwählen. Diese WebControlls werden in Form von Java-Klassen in das V4Servlet implementiert und stellen grafische Javainteraktionselemente dar.

Bei der Oberflächenentwicklung zu der Seite „Vertrag suchen“ sind verschiedene Funktionalitäten unverzichtbar. So muss ein Feld angeboten werden, aus welchem der Benutzer ein entsprechendes Suchkriterium auswählen kann und ein weiteres Feld, in welches ein Suchbegriff eingegeben wird. Die vom System zu den Eingaben ermittelten Daten werden in einem weiteren Feld aufgezeigt und sind von dort weiter

verwendbar. Mehrere Verweise bieten entsprechende Funktionen oder Weiterleitungen zu anderen Seiten an. Die Programmierung des V4Servlets dieser Seite wird im Folgenden in einzelnen Schritten dargestellt, wobei der gesamte Quellcode der Web-Applikation des License-Management-System in der beiliegenden CD zur Verfügung gestellt wird.

```
1 public class Vertrag_Suchen extends V4Servlet {
2     public void v4Main(User user, HttpServletRequest req,
3                        HttpServletResponse res) throws IOException {
4
5         Page page = new Page(req, res);
6         JScript jscript1 = new JScript("../Vertrag_Suchen.js");
7         page.addJScript(jscript1);
8
9         Tray tray1 = new Tray();
10        tray1.setTitle("Vertrag suchen");
11        tray1.setWidth("900");
12        tray1.setHeight("400");
13        tray1.setTooltip("Tooltip tray1");
14        GridLayout grid1 = new GridLayout();
15        GridLayoutCell cell;
16
17        // fehlende implementierte Web-Controlls
18
19        tray1.setContent(grid1);
20        page.addContent(tray1);
21        page.write();
22    }
23}
```

Abbildung 7-2: Standard-V4Servlet-Containers

Der Grundaufbau dieser Seite ist exemplarisch für den Aufbau aller Seiten dieser Web-Applikation. In Zeile 1 wird das V4Servlet mit der Klasse „Vertrag_Suchen“ generiert. Dieses beinhaltet in Zeile 2 die Methode v4Main, welche die Benutzeranfragen (HttpServletRequest req) auswertet und die dazugehörigen Antworten (HttpServletResponse res) ausgibt. In dieser Methode werden die Interaktionselemente (WebControlls) implementiert.

Um WebControlls in der v4Main-Methode anzuordnen, ist es notwendig, ein neues PageObjekt zu erzeugen, siehe Zeile 5.

Als Nächstes wird in Zeile 6 – 7 ein JScript-Objekt gebildet, welches den Zugang zu der Controller-Methode durch den übergebenen Klassenpfad ermöglicht. Dies wird der neu erzeugten Seite (page) zugeordnet.

Auf dem Tray in den Zeilen 9 -13 werden die Interaktionselemente der sichtbaren Views durch das neu erzeugte GridLayout-Objekt erzeugt (Zeile 14- 15) und in seinem Erscheinungsbild genau definiert.

Anschließend wird ab Zeile 19 das GridLayout-Objekt dem Tray-Objekt zugewiesen, dieses der Seite, die abschließend dem Anwender eine HTML-Seite generiert und darstellt. Dieses grobe Gerüst der Seite „Vertrag suchen“, welches für alle weiteren

Views gleich ist, muss im Folgenden noch mit spezifischen WebControlls gefüllt werden, die dem Anwender diverse Eingabemöglichkeiten zur Verfügung stellt:

```
1 TrayCommand tc0 = new TrayCommand("tc0", "Suchen", "getVertrag()");
2 tc0.setTooltip("Tooltip tc0");
3 tray1.addTrayCommand(tc0);
4
5 TrayCommand tc1 = new TrayCommand("tc1", "Vertrag entfernen",
6                                "setEntfernenData()");
7 tc1.setTooltip("Tooltip tc1");
8 tray1.addTrayCommand(tc1);
9
10 TrayCommand tc3 = new TrayCommand("tc3", "Abbrechen",
11                               "refresh_vertragSuchen()");
12 tc3.setTooltip("Tooltip tc3");
13 tray1.addTrayCommand(tc3);
14
15 TrayCommand tc4 = new TrayCommand("tc4", "Start",
16                         "setSubtray('com.dcx.lms_test.Start')");
17 tc4.setTooltip("Tooltip tc4");
18 tray1.addTrayCommand(tc4);
```

Abbildung 7-3: WebControl „TrayCommand“ in Java-Syntax

Wird ein TrayCommand betätigt, wird die vom Anwender gewünschte JavaScript-Funktion ausgeführt. Wird beispielsweise das TrayCommand „Suchen“ vom Anwender betätigt, werden die entsprechenden Daten an die JavaScript-Funktion „getVertrag()“ übergeben, welche wiederum das V4RPCServlet aufruft, welche die Daten in der Datenbank sucht und das Ergebnis zurück liefert.

Neben diesen Funktionen wird im weiteren Verlauf ein Dropdown-Feld in den Quellcode des V4Servlets implementiert, der verschiedene Suchkriterien zur Verfügung stellt:

```
1  DropdownListBox dd1 = new DropdownListBox("dd1");
2  dd1.setWidth("180");
3  dd1.setValue("Sachnummer");
4
5  ListBoxItem item1;
6  ListBoxCell eintrag;
7  item1 = new ListBoxItem("b.sachnummer");
8  item1.addListBoxCell(new ListBoxCell("Sachnummer"));
9  dd1.addListBoxItem(item1);
10
11 ListBoxItem item2;
12 item2 = new ListBoxItem("vk.vertragskategoriebezeichnung");
13 item2.addListBoxCell(new ListBoxCell("Vertragsart"));
14 dd1.addListBoxItem(item2);
15
16 ListBoxItem item3;
17 item3 = new ListBoxItem("va.vertragsname");
18 item3.addListBoxCell(new ListBoxCell("Vertragsname"));
19 dd1.addListBoxItem(item3);

20 ListBoxItem item4;
21 item4 = new ListBoxItem("p.produktname");
22 item4.addListBoxCell(new ListBoxCell("Produktname"));
23 dd1.addListBoxItem(item4);
```

Abbildung 7-4: WebControl „Dropdown-Listbox“

Die hier dargestellte Dropdown-Listbox stellt dem Anwender diverse Suchkriterien wie „Sachnummer“, „Vertragsart“ oder „Produktname“ zur Auswahl. In den Zeilen 1-3 wird ein neues Dropdown-Listbox-Objekt erzeugt und genauer definiert. Die einzelnen ListBoxItems werden in den Zeilen 5-23 festgelegt und können vom Anwender in der View durch Anklicken ausgewählt werden.

Ein weiteres Web-Control ist das InputField, in welches der Anwender den individuellen Suchbegriff eingeben kann:

```
1  InputField field1 = new InputField("field1");
2  field1.setRequired(false);
3  field1.setValue("");
4  field1.setWidth("170");
```

Abbildung 7-5: WebControl „InputField“

Das InputField wird in Zeile 1 neu erzeugt und steht zur Eingabe dem Anwender zur Verfügung.

Sind die WebControlls „Suche nach“ und „Suchbegriff“ ausgefüllt und das TrayCommand „Suchen“ wurde betätigt, werden die ermittelten Ergebnisse in der ListBox ausgegeben.

```
1 ListBox lb1 = new ListBox("lb1");
2 lb1.setModel("getVertrag@com.dcx.lms_test.data.Vertrag_SuchenData");
3
4 ListBoxHeader listboxheader1 = new ListBoxHeader();
5
6 listboxheader1.addListBoxHeaderCell(
7      new ListBoxHeaderCell("Sachnummer"));
8 listboxheader1.addListBoxHeaderCell(
9      new ListBoxHeaderCell("Vertragsart"));
10 listboxheader1.addListBoxHeaderCell(
11     new ListBoxHeaderCell("Vertragsname"));
12 listboxheader1.addListBoxHeaderCell(
13     new ListBoxHeaderCell("Hersteller"));
14 listboxheader1.addListBoxHeaderCell(
15     new ListBoxHeaderCell("Produkt"));
16 listboxheader1.addListBoxHeaderCell(
17     new ListBoxHeaderCell("Version"));
18 listboxheader1.addListBoxHeaderCell(
19     new ListBoxHeaderCell("Laufzeit"));
20 listboxheader1.addListBoxHeaderCell(
21     new ListBoxHeaderCell("Beschaffungsdatum"));
22 lb1.setListBoxHeader(listboxheader1);
```

Abbildung 7-6: WebControl „ListBox“

Diese ListBox gibt die zu den Suchbegriffen ermittelten Daten aus. Dabei werden verschiedene Informationen der Daten ausgegeben, wie in den Zeilen 6-7 die „Sachnummer“ oder in den Zeilen 16-17 die „Version“. Unter diesen Überschriften im ListBox-Header werden die gesuchten Daten aufgeführt und stehen nun dem Anwender zu weiteren Operationen zur Verfügung, wie sie bei den TrayCommands

beschrieben wurden. Das RPCMessage-Objekt stellt dabei die Verbindungen zwischen den einzelnen MVC-Bereichen her und übermittelt die eingegebenen Daten.

Diese Web-Controlls müssen dem GridLayout mit ihrem Objektnamen zugewiesen werden. Das GridLayout wird benutzt zur Anordnung der Web-Controlls in dem bereits beschriebenen Tray. Ein GridLayout wird hier als eine Tabelle mit Zeilen und Spalten definiert, in deren Tabellenzellen die Web-Controlls eingebunden werden. Da diese Einbindung in der Umsetzung sehr umfangreich ist, wird hier nur beispielhaft die Einbindung einer einzelnen Zeile in das Tray dargestellt:

```
1 cell = new GridLayoutCell(grid1.getNextRowIndex(), 1);
2 grid1.addGridLayoutCell(cell);
3 cell.setWidth("25");
4 cell = new GridLayoutCell(grid1.getCurrentRowIndex(), 2,label1);
5 grid1.addGridLayoutCell(cell);
6 cell.setColSpan(2);
7 cell = new GridLayoutCell(grid1.getCurrentRowIndex(), 3, dd1);
8 grid1.addGridLayoutCell(cell);
9 cell.setColSpan(2);
10 cell = new GridLayoutCell(grid1.getCurrentRowIndex(), 4);
11 grid1.addGridLayoutCell(cell);
12 cell.setWidth("50");
13 cell = new GridLayoutCell(grid1.getCurrentRowIndex(), 5, label2);
14 grid1.addGridLayoutCell(cell);
15 cell.setColSpan(2);
16 cell = new GridLayoutCell(grid1.getCurrentRowIndex(), 6, field1);
17 grid1.addGridLayoutCell(cell);
18 cell.setWidth("100");
19 cell = new GridLayoutCell(grid1.getCurrentRowIndex(), 7);
20 grid1.addGridLayoutCell(cell);
21 cell.setWidth("25");
```

Abbildung 7-7: Zuweisung individueller WebControlls an das GridLayout

Hier werden nun die bereits definierten WebControlls label1, dd1, label2 und field2 dem GridLayout-Objekt zugewiesen. Dieses GridLayout-Objekt wird dem Tray-Objekt übergeben, welches wiederum dem Page-Objekt zugeteilt wird. Dieses Page-Objekt wird letztendlich in HTML-Code kompiliert und im Browser dem Anwender dargestellt.
Die aus den aufgeführten Quellcodekomponenten entwickelte View ergibt eine Ein-/Ausgabe-Maske, mit welcher der Anwender mithilfe bestimmter Suchkriterien nach Verträgen suchen kann, die im aktiven Datenbestand des License-Management-System gespeichert sind. Die aufgefundenen Daten können für weiterführende Operationen genutzt werden. Eine genauere Beschreibung dieser und aller anderen Masken mit ihren Funktionen des License-Management-System wird im Anhang unter „Anwenderhandbuch“ zur Verfügung gestellt.
Die View „Vertrag suchen“ wird in der folgenden Abbildung dargestellt:

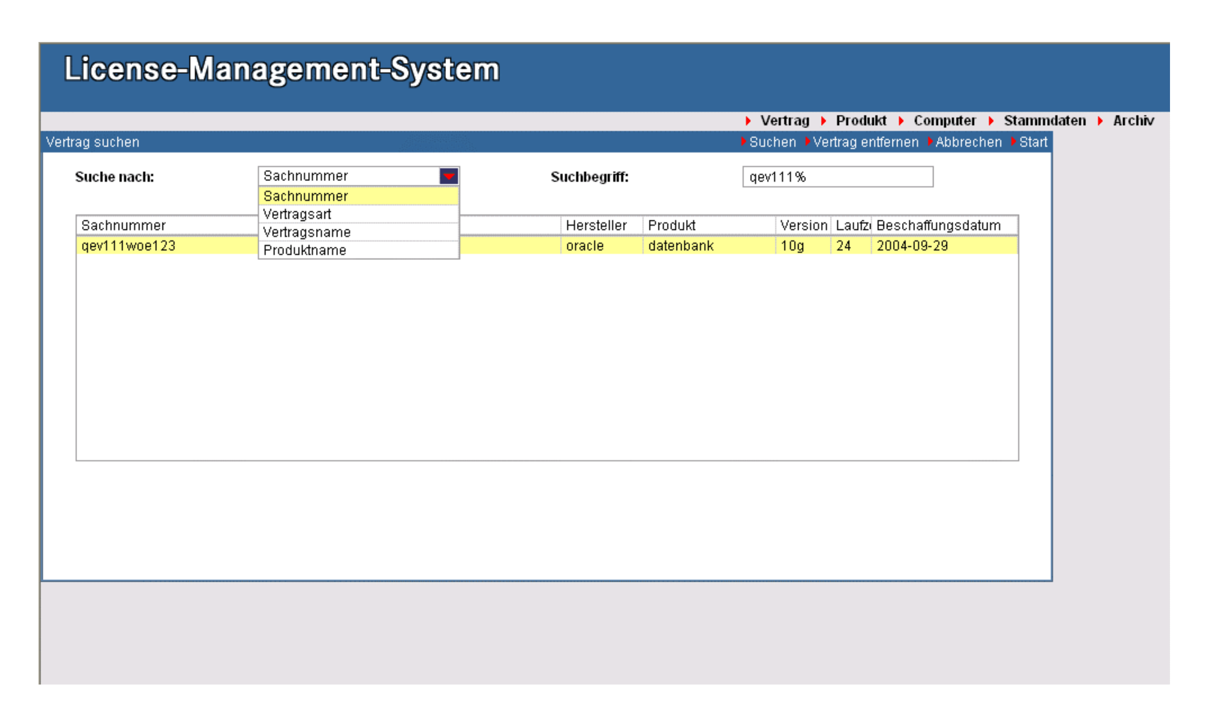


Abbildung 7-8: View der Funktion „Vertrag suchen“

7.2.2. Entwicklung eines Controllers am Beispiel “Vertrag suchen”

Controller sind für die Ablaufsteuerungen zwischen View und Model zuständig. Es bestimmt, welches View-Ereignis welche Model-Methode aufruft.

Der Controller wird in JavaScript entwickelt, wobei JavaScript eine Erweiterung des HTML-Codes in Form von Klartext ist. Da JavaScript lediglich eine Erweiterung des HTML-Codes darstellt, ist JavaScript als eingebundener Bestandteil eines HTML-Gerüsts zu verwenden. Der Vorteil der Programmierung des Controllers in JavaScript ist, dass der Controller mithilfe des JavaScript spezielle Remote-Methoden ausrufen kann und die Daten in der View auf Korrektheit überprüft. Von Vorteil ist bei der Verwendung von JavaScript, dass der beschriebene Vorgang erheblich beschleunigt wird, da der Browser keine Anfrage (Request) an den Server schicken und die Antwort (Response) nicht abwarten muss.

Der mithilfe der JavaScript-Methoden entwickelte Controller übernimmt die bereits beschriebenen Daten aus der View „Vertrag suchen“ und leitet diese an das Model weiter. Er nimmt eine Vermittlerposition ein und sorgt für einen problemlosen Datentransfer. Die Entwicklung eines solchen Controllers wird am Beispiel „Vertrag suchen“ aufgezeigt, wobei hier die Funktion „getVertrag()“ beispielhaft dargestellt wird:

```
1 function getVertrag() {
2     var msg = new RPCMessage();
3     msg.addString("valueInputField", field2.getValue());
```

```
4     msg.addString("valueListBox", dd1.getValue());
5     listbox1.setRPCMessage(msg);
6 }
```

Abbildung 7-9: JavaScript-Funktion „getVertrag()“ in der Rolle eines Controllers

Wird auf dem View „Vertrag suchen“ das TrayCommand „Suchen“ betätigt, holt sich die Funktion „getVertrag()“ aus Zeile 1-7 die eingegebenen Werte der DropDown-Listbox, sowie des Input-Fields. Diese Daten werden den Variablen „valueInputField“ und „valueListBox“ aus den Zeilen 3 und 4 zugewiesen. Um die Daten an das Model übergeben zu können, ist es notwendig, ein neues RPCMessage-Objekt wie in Zeile 1 zu erstellen und diesem die Werte der Variablen aus Zeile 3-4 zuzuordnen. Das WebControl ListBox mit der Bezeichnung „listbox1“ bekommt die Werte mithilfe der Methode „setRPCMessage(msg)“ zugewiesen. Das ListBox-Objekt springt mithilfe der Methode „listbox1.setRPCMessage(msg)“ zum Model „Vertrag_SuchenData“, welches das RPCMessage-Objekt zur weiteren Verarbeitung übernimmt. Dies wird ermöglicht durch die Implementierung der Methode „listbox1.setModel()“ in der View „Vertrag suchen“, welche den Pfad zu dem Modell in der Zeile 2 definiert hat.
Die weitere Bearbeitung der Daten im Model wird im folgenden Unterkapitel beschrieben.

7.2.3. Entwicklung eines Model am Beispiel “Vertrag suchen”

Für die Entwicklung eines Models wird das V4RPCServlet benötigt. Dies ist eine Erweiterung des Standart-HTTPServlets und bekommt ein User- und ein RPCMessage-Objekt übergeben. Die Methoden des V4RPCServlets werden durch den Remote Procedure Call (RPC) in der View oder im Controller aufgerufen und geben bestimmte Werte weiter. Die übergebenen Werte werden in diesem Servlet entgegengenommen, verarbeitet und die Ergebnisse der aufrufenden Funktion mithilfe des RPCMessage-Objekts wieder zurückgegeben.
Um diese Ergebnisse zu bekommen, muss auf die Datenbank des License-Management-System zugegriffen werden. Java Database Connectivity (JDBC) ermöglicht diesen Zugriff auf eine Datenbank.

```
1 public class Vertrag_SuchenData extends V4RPCServlet {
2     public RPCMessage getVertrag(User user, RPCMessage msg) {
3
4         ListBoxModel model = new ListBoxModel(msg);
5         ListBoxItem item;
6         ListBoxCell cell;
7
8         String valueInputField = msg.getString("valueInputField");
```

```
9          String valueDDListBox = msg.getString("valueListBox");
10
11         // fehlende Datenbankanfrage
12         return model.getMessage();
13      }
14 }
```

Abbildung 7-10: V4RPCServlet-Container

Diese Abbildung zeigt das Grundgerüst eines Models, wobei hier in den Zeilen 4-9 spezifische Angaben zu dem Model „Vertrag suchen“ gemacht werden. Die Klasse „Vertrag_SuchenData“ des V4RPCServlets beinhaltet die öffentliche Methode „getVertrag()“, welche ein User-Objekt und ein RPCMessage-Objekt als Parameter übergeben bekommen. Der Rückgabetyp der Methode „getVertrag“ ist ebenfalls ein RPCMessage-Objekt, wie auch in der „return“-Anweisung in Zeile 12 zu erkennen ist. Dabei muss zuerst eine Datenbankanfrage gestartet werden, bevor die Ergebnisse dieser Anfrage zurückgegeben werden können. Diese Datenbankanfrage muss in den beschriebenen Vorgang mit eingebunden werden, wie in Zeile 11 markiert ist. Der Aufbau einer solchen Datenbankanfrage wird in der nächsten Abbildung erläutert:

```
1  Database db = new Database("lms");
2
3  try {
4      db.open();
5      Statement stmt = db.createStatement();
6      ResultSet rs = stmt.executeQuery( +
7            "SELECT b.sachnummer, " +
8            "       vk.vertragskategoriebezeichnung, " +
9            "       va.vertragsname, " +
10           "       h.herstellername, " +
11           "       p.produktname, " +
12           "       vs.version_nr, " +
13           "       v.laufzeit, " +
14           "       b.beschaffungsdatum " +
15           "FROM vertrag v " +
16           "INNER JOIN beschaffung b " +
17           "ON b.beschaffungsdatum = v.beschaffungsdatum " +
18           "AND b.sachnummer = v.sachnummer " +
19           "INNER JOIN vertragsart va " +
20           "ON v.vertragsart_nr = va.vertragsart_nr " +
21           "INNER JOIN vertragskategorie vk " +
22           "ON v.vertragskategorie_nr = vk.vertragskategorie_nr " +
23           "INNER JOIN produkt p " +
24           "ON p.vertrag_nr = v.vertrag_nr " +
25           "INNER JOIN hersteller h " +
26           "ON p.hersteller_nr = h.hersteller_nr " +
27           "INNER JOIN version vs " +
28           "ON vs.produkt_nr = p.produkt_nr " +
29           "WHERE " + valueDDListBox + " LIKE LCASE ('" +
30           valueInputField + "')")
31
32     while(rs.next()) {
33         item = new ListBoxItem(rs.getString("sachnummer"));
34         item.addListBoxCell(new
35         ListBoxCell(rs.getString("sachnummer")));
36         item.addListBoxCell(new
37         ListBoxCell(rs.getString("vertragskategoriebezeichnung")));
38         item.addListBoxCell(new
39         ListBoxCell(rs.getString("vertragsname")));
```

```
40          item.addListBoxCell(new
41          ListBoxCell(rs.getString("herstellername")));
42          item.addListBoxCell(new
43          ListBoxCell(rs.getString("produktname")));
44          item.addListBoxCell(new
45          ListBoxCell(rs.getString("version_nr")));
46          item.addListBoxCell(new
47          ListBoxCell(rs.getString("laufzeit")));
48          item.addListBoxCell(new
49          ListBoxCell(rs.getString("beschaffungsdatum")));
50          model.addListBoxItem(item);
51      }
52      rs.close();
53      stmt.close();
54
55      } catch(SQLException e) {
56            logger.error(e);
57            msg.addString("error", "Es ist ein Fehler aufgetreten!");
58      } finally {
59            db.close();
60 }
```

Abbildung 7-11: JDBC-Funktionalität im v4RPCServlet

Um die Datenverbindung zu der in DB2 implementierten Datenbank des License-Management-System aufzubauen, wird das Entwicklungstool WebSphere-Studio-Application-Developer (WSAD) genutzt. Mit diesem ist es möglich, Web-Applikationen zu entwickeln. Den Datenbankverbindungsaufbau übernimmt der WSAD und erleichtert so den Zugriff auf die Datenbank. Hier muss, wie in Zeile 1 aufgezeigt, ein Datenbankobjekt erstellt werden. Das Laden des JDBC-Treibers für das Datenbank-Management-System DB2 sowie die Herstellung der Verbindung zur Datenbank mittels eines Connection-Objekt übernimmt der WebSphere Studio Application Developer.

Das Öffnen der Datenbank wird in der Zeile 4 gezeigt. Nachdem die Datenbank geöffnet ist, können individuelle Anfragen an diese gestellt werden. Diese Anfragen werden in Form von SQL-Statements, die die Ausführung von SQL-Anweisungen über gegebene Verbindungen ermöglichen, in der Methode „stmt.executeQuery()“ als Parameter implementiert. Diese SQL-Anfrage wird an die Datenbank gesendet. Die Datenbank verarbeitet diese Anfragen und gibt das Ergebnis in Form eines ResultSet an das V4RPCServlet zurück. Das ResultSet „verwaltet die Ergebnisse einer Anfrage in Form einer Relation und unterstützt den Zugriff auf einzelne Spalten.“ Die erhaltenen Ergebnisse werden in einer while-Schleife den ListBox-Items zugewiesen. Wenn keine Ergebnisse mehr im ResultSet vorhanden sind und so zugewiesen werden können, werden das ResultSet und anschließend das Statement-Objekt geschlossen.

Ist die Abfrage beendet und die Ergebnisse im ResultSet den ListBoxItems zugewiesen, werden die benutzten Ressourcen durch Aufruf der „close“-Methoden von ResultSet und Statement frei gegeben, siehe Zeilen 52 und 53. Dies hat den Vorteil, dass sofort die Ressourcen des Datenbank-Management-Systems wieder frei gegeben

und Speicherplatzprobleme vermieden werden. In Zeile 59 wird abschließend die Datenbankverbindung selbst geschlossen.

Die ermittelten Daten werden, wie in Abbildung 7-10 Zeile 12 dargestellt, an das aufrufende Servlet mit dem RPCMessage-Objekt an die View zurückgegeben, welche diese Ergebnisse in der ListBox darstellt.

Somit ist die Abfrage, die in diesem Fall die Funktion „Vertrag suchen" beinhaltete, beendet und die gesuchten Daten werden dem Anwender in der ListBox aufgelistet und stehen von dort zu weiteren Verarbeitungen zur Verfügung.

Der Prototyp des License-Management-System ermöglicht in der beschriebenen Funktion die Suche nach bestimmten Verträgen. Diese Funktion kann als Beispiel für die weiteren Masken gesehen werden, die im Anhang und auf der beiliegenden CD näher ausgeführt werden. Die Entwicklung dieser Seiten orientiert sich immer an den drei Schritten des MVC-Entwurfsmusters, die hier im Einzelnen erläutert wurden. Mithilfe des MVC-Entwurfsmusters ist es möglich, viele der in der „Informationsanforderung" und im weiteren Verlauf der Diplomarbeit genannten Anforderungen an das License-Management-System zu verwirklichen. Die Umsetzung solcher Anforderungen mit all ihren Verknüpfungen wurde hier exemplarisch dargestellt und bietet letztendlich eine Eingabe-/Ausgabemaske, mit welcher auf Informationen in der Datenbank zugegriffen werden kann und die so gefundenen Informationen dem Anwender zur Verfügung gestellt werden.

8. Fazit und Ausblick

Das Ziel dieser Diplomarbeit war die Konzeption und Realisierung eines Prototyps für ein License-Management-System. Das entwickelte License-Management-System wurde speziell auf die Bedürfnisse und Strukturen des Unternehmens DaimlerChrysler am Standort Wörth abgestimmt.

Zur Erreichung dieses Ziels wurde neben allgemeinen Informationen zu bestehenden Lizenzverträgen und Volumenlizenzmodellen die momentane Vorgehensweise der Lizenzbeschaffung und Lizenzverwaltung im Unternehmen DaimlerChrysler untersucht. Der Untersuchung dienende interne Informationen zu den Lizenzverwaltungssystemen bei DaimlerChrysler am Standort Wörth wurden gesammelt, abgebildet und ausgewertet. Am Ende der Analyse waren konkrete Probleme in der aktuellen Lizenzverwaltung offensichtlich, wie der Verlust von Lizenzen beim Entfernen eines Computers. Diese Probleme entstehen in erster Linie aus einer mangelnden Verwaltung der Softwarelizenzen, die wiederum einen unzureichenden Überblick über gekaufte und disponible Lizenzen mit sich bringt. Um diese Schwierigkeiten weitestgehend durch einen Prototyp zu lösen, wurden im nächsten Schritt alle modifizierten Anforderungen und Funktionalitäten an das optimierte License-Management-System ausführlich dargestellt. Die Informationsanforderungen wurden in verschiedene Kernbereiche aufgeteilt und nachfolgend die Funktionsanforderungen erstellt in welchen die Komplexität der Funktionen und die Verknüpfungen der einzelnen Bereiche untereinander aufgezeigt wurden. Im nächsten Schritt wurden diese komplexen Verknüpfungen der Lizenzverwaltung in einem abstrakten Modell zusammengefasst und anschaulich gemacht. Die optimale Umsetzung wird durch das Entity-Relationship-Modell gewährleistet, denn Eigenschaften wie Aussagekräftigkeit oder Einfachheit schaffen einen schnellen und klar strukturierten Überblick über die gegebenen Verhältnisse. Diese klare Struktur schützte dennoch nicht vor einigen Problemen bei der Erstellung des ER-Modells. Es kam zu Abgrenzungsschwierigkeiten der Kernbereiche, deren Überschneidungen vor allem bei der Verfeinerung der Entitäten deutlich wurden. Außerdem musste bei den Basisinformationen zu den verschiedenen Themen dieser Diplomarbeit häufig stark selektiert und lediglich für den Sachverhalt relevante Informationen genannt werden. Nach der Fertigstellung des ER-Modells und dessen Beschreibung wurde dieses Modell folgend in ein relationales Datenbankschema gebracht. Dieses Datenmodell wurde mithilfe der Datendefinitionssprache der relationalen Datenbanksprache SQL in das relationale Datenbank-Management-System DB2 Version 8 vom Unternehmen IBM implementiert.

Das relationale Datenbank-Management-System ist bei DaimlerChrysler am Standort Wörth ein strategisches Produkt, das der Elaboration dieser Diplomarbeit zur Seite gestellt wurde. Die Umsetzung des ER-Modells in das relationale Modell wurde exemplarisch an einer Funktion ausführlich behandelt, alle weiteren Funktionen können äquivalent dazu transferiert werden. In einem letzten Schritt wurde mithilfe des Model-View-Controller-Entwurfsmusters eine Web-Applikation für das License-Management-System entworfen. Die Erstellung der grafischen Benutzeroberfläche und die Verknüpfung zur Datenbank ermöglichen es dem Anwender, mithilfe dieser Eingabe-Ausgabe-Masken auf die Datenbank zuzugreifen und die gewünschten Operationen auszuführen. Diese Web-Applikation wurde im V4-Framework vorgenommen, ein auf das Unternehmen DaimlerChrysler spezialisiertes Framework. Die exklusive Anwendung in diesem Unternehmen brachte es mit sich, dass nur wenig allgemeine Literatur zu diesem Framework verfügbar war. Trotz dieser Schwierigkeiten konnte durch die Benutzung der gegebenen Web-Controlls des Frameworks eine gleichmäßige Struktur der Web-Applikation erreicht werden. Die einfache Wartung und die Konsistenz der Daten wurde durch die Nutzung des MVC-Entwurfsmusters erreicht. Der so entwickelte Prototyp erfüllt sehr viele der Anforderungen, die ursprünglich gesammelt und an das optimale License-Management-System gestellt wurden. Viele Funktionen sind leichter auszuführen als zuvor und es ergibt sich ein besserer Überblick über die gekauften Softwarelizenzen und die implizierten Produkte, die dem Unternehmen eine gute Wettbewerbsfähigkeit und gesteigerte Produktivität gewährleisten. Der Prototyp bietet einen umfassenden Überblick über den Lizenzverkehr und ermöglicht eine einfache Wartung und Handhabung der Lizenzverwaltung.

Zu beachten ist, dass die Ausarbeitung eines Prototyps nicht gleichbedeutend ist mit der Erstellung eines fertigen und voll ausgereiften Programms, sondern das Grundgerüst eines solchen darstellt. Die entworfene Grundstruktur kann durch Erweiterung der Datenbank und ausführliche Tests zur Nutzung freigegeben werden.

Die Diplomarbeit bietet Basisinformationen zum Thema und stellt alle im Unternehmen gesammelten Informationen zur Verfügung. Sie gibt einen umfassenden Überblick über alle Aspekte der Entstehung eines Prototyps und die Anforderungen an ein License-Management-System. Es wird aufgezeigt, wie bereits bestehende und genutzte Beschaffungswege von Lizenzen analysiert und optimiert werden. Auch wenn viele Anforderungen an ein solches System umgesetzt wurden, ist eine weitere Verfeinerung in einigen Bereichen möglich. Die Umsetzung der Funktion „Report“ wurde beispielsweise nicht umgesetzt, da das Framework V4 keine geeignete Methode zur Implementierung der Reportfunktion zur Verfügung stellt. Es wäre denkbar, dem

License-Management-System eine Funktion hinzuzufügen, die automatisch auf neu erschienene Software oder Updates hinweist. Auch wäre eine Funktion möglich, welche die im aktiven Datenbestand genutzten Computer auf den Gebrauch von illegaler und nicht lizenzierter Software überprüft und das Unternehmen dadurch vor illegalen Handlungen schützt. Eine standortübergreifende Nutzung des License-Management-System bietet einen Austausch und die Grundlage zum kostengünstigeren Lizenzeinkauf. Eine weiter angestrebte Funktion ist die Kombination des License-Management-System mit dem Software Management System des Unternehmens Microsoft. Eine Kopplung beider Systeme offerierte ein automatisches Update des aktiven Datenbestandes bei Vergabe von Produktlizenzen an Computer durch das Software Management System.

Grundsätzlich gilt, dass die Nutzung von Software quantitativ zunimmt und damit der Stellenwert von Softwarelizenzen zukünftig steigt. Steigende Investitionen und Abhängigkeit führen zu einer stärkeren Fokussierung der Unternehmen auf die Lizenzverwaltung. Diesen hohen inneren Erwartungen kombiniert mit externen Veränderungen muss das License-Management-System gerecht werden.

9. Literaturverzeichnis

[bsa] Business Software Alliance,
http://global.bsa.org,
letzter Zugriff: 25.09.2004

[bsa] Business Software Alliance,
http://global.bsa.org/germany/piraterie/konsequenzen.php,
letzter Zugriff: 25.09.2004

[bsa] Business Software Alliance,
http://global.bsa.org/germany/piraterie/konsequenzen.php
letzter Zugriff: 25.09.2004

[bsa] Business Software Alliance,
http://global.bsa.org/germany/piraterie/warum.php
letzter Zugriff: 25.09.2004

[bsa] Business Software Alliance für Deutschland. Leitfaden für Softwaremanagement S.13,
http://global.bsa.org/germany/info/pics/guide.d.pdf
letzter Zugriff: 25.09.2004

[bue03] Bürkner, Reimer M.: Erfolgreiche Software-Lizenzierung. Electronic License Management , Springer-Verlag, Berlin, Heidelberg 2003.

[che76] Chen, P.: The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems, Band 1, Nr. 1, S. 9-36, 1976.

[elml02] Elmasri, Ramez / Navathe, Shamkant B.: Grundlagen von Datenbanksystemen, 3., überarb. Aufl., München 2002, Pearson Studium, S. 576.

[gam98] Gamma, Erich / Helm, Richard / Johnson, Ralph / Vlissides, John: *Entwurfsmuster,* Addison - Wesley, 1998.

[heu00] Heuer, Andreas / Saake Gunter: Datenbanken: Konzepte und Sprachen, 2. Aufl., Bonn 2000, MITP-Verlag.

[saa00] Saake, Gunter / Sattler, Kai-Uwe: Datenbanken & Java. JDBC, SQLJ und ODMG, 1. Aufl., Heidelberg 2000, S. 56.

[sau98] Sauer, Hermann: Relationale Datenbanken – Theorie und Praxis. Mit einem Beitrag zu SQL-3 von Klaus Grieger, 4. akt. u. erw. Aufl., Bonn, München, u.a. 1998, S. 36.

[sau98] Sauer, Hermann: Relationale Datenbanken – Theorie und Praxis. Mit einem Beitrag zu SQL-3 von Klaus Grieger, 4. akt. u. erw. Aufl., Bonn, München, u.a. 1998, S. 224.

[sau98] Sauer, Hermann: Relationale Datenbanken – Theorie und Praxis. Mit einem Beitrag zu SQL-3 von Klaus Grieger, 4. akt. u. erw. Aufl., Bonn, München, u.a. 1998, S. 225.

[sau98] Sauer, Hermann: Relationale Datenbanken – Theorie und Praxis. Mit einem Beitrag zu SQL-3 von Klaus Grieger, 4. akt. u. erw. Aufl., Bonn, München, u.a. 1998, S. 225.

[sta02] Starke, Gernot: Effektive Software-Architekturen. Ein praktischer Leitfaden, München, Wien 2002, S. 159.

[symantec] Symantec Value Program,
http://enterprisesecurity.symantec.de/PDF/LicProg_EndUser-value-GE.pdf?EID=0
letzter Zugriff: 25.09.2004

[symantec] Symantec Elite Program,
http://enterprisesecurity.symantec.de/PDF/LicProg_EndUser-Elite-GE.pdf?EID=0
letzter Zugriff: 25.09.2004

[symantec] Symantec Programm-Broschüre,
http://enterprisesecurity.symantec.de/PDF/overview_brochure_DE.pdf?EID=0
letzter Zugriff: 25.09.2004

[zeh89] Zehnder, Carl August: Informationssysteme und Datenbanken, 5. durchges. Aufl., (Leitfaden der angewandten Informatik) Stuttgart 1989, S. 23.

10. Anhang

10.1. *Allgemein vorkommende Lizenzierungsarten*

Bei der Betrachtung der verschiedenen Lizenzierungsarten ist augenfällig, dass die von den großen Lizensierungstoolherstellern angebotenen Lizenzarten sehr vielfältig und somit sehr spezifisch sind, was der Befriedigung der jeweiligen Kundenbedürfnisse sehr entgegenkommt. So können die Lizenzen je nach Lizenzierungsart an ein bestimmtes Produkt, eine bestimmte Hardware oder aber an einen bestimmten Benutzer gebunden sein.
Wegen der Vielzahl der verschiedenen Softwarehersteller ist es nur schwer möglich, die jeweiligen Lizenzierungsarten darzustellen, doch die folgende Aufstellung der zur Zeit marktüblichen Lizenzierungsarten, die in weiten Teilen dem Buch „Erfolgreiche Software-Lizenzierung“ von Reimer Bürkner [bue03] entnommen ist, soll einen Einblick in diese vielseitige Thematik geben und die Vielfältigkeit der möglichen Lizenzierungsarten herausstellen.

Demo (Evaluierungs-Lizenz)
Diese Lizenz wird auch TBYB, Try-Before-You-Buy, genannt und diese Bezeichnung gibt bereits Aufschluss über die Besonderheit dieser Lizenzierungsart. Denn mit der Demo-Version wird eine Lizenz nur zeitlich begrenzt nutzbar zur Produktevaluation. Der Kunde wird so an ein Produkt herangeführt und erhält die Möglichkeit, dieses zu testen, bevor er es kauft. Dabei kann die Funktionalität dieses Produktes gegenüber der Vollversion eingeschränkt sein. Die zeitliche Beschränkung kann durch eine bei der Lizenzerstellung festgelegte Gültigkeitsdauer definiert werden, oder die Lizenz erlischt n Tage nach der Installation oder nachdem die Anwendung m mal aufgerufen wurde. Danach kann der Kunde entscheiden, ob und in welcher Anzahl er die Lizenz als Vollversion kaufen möchte.

Globale Gültigkeit (run-anywhere)
Die lizenzierte Software läuft auf jedem System.

Enable/Disable Product-Features
Hier ist es möglich, durch viele lizenzierbare Leistungsmerkmale, so genannte Features, ein Produkt zu definieren. Dadurch müssen gemeinsame Leistungsmerkmale für verschiedene Produkte nicht einzeln lizenziert werden, sonder dies kann

gleichzeitig geschehen. So entsteht die Möglichkeit, zusammengehörige aber nicht identische Produkte zusammen zu lizenzieren, wie beispielsweise eine „Light"-Version mit nur eingeschränkter Funktionalität, eine „Standard"-Version für den üblichen Gebrauch und eine „Professional"-Version mit noch größerer Leistungsfähigkeit und Funktionalität.

Floating (concurrent) in einem Netzwerk

Diese Lizenzierungsart ist der klassische Fall netzwerkfähiger und gleichzeitig, also concurrent, genutzter Lizenzen. Hierbei ist aber festgelegt, dass nie mehr als n Lizenzen zur gleichen Zeit im Netzwerk genutzt werden dürfen.

Nutzerbezogen (persönliche Lizenzen)

Diese Lizenz, die an den Login-Namen in einem Netzwerk gekoppelt ist, erlaubt, dass die Software auf jedem Rechner des Netzwerkes genutzt werden kann, wenn der Login-Name in der erlaubten Nutzerliste enthalten ist. Dabei darf gewöhnlich nicht mehr als ein Rechner mit dem gleichen Namen die Lizenz nutzen. Die Anzahl der lizenzierten Benutzer wird durch den Software-Lieferanten definiert, der Systemadministrator oder der Lieferant spezifizieren die Benutzernamen.

Hardwaregebunden (node locked)

Mit dieser Lizenzierungsart wird die Software auf einen bestimmten Rechner und für einen unbegrenzten Zeitraum lizenziert. Die Hardwarebindung wird mithilfe der MAC-Adresse, Dongle-ID oder der Seriennummer der Festplatte festgelegt.

Hardwaregebunden, anzahl-begrenzt (counted)

Die Software wird für den gleichzeitigen n-maligen Gebrauch auf einem einzelnen Computersystem lizenziert.

Pakete (packages)

Bei diesen Paketen wird eine bestimmte Menge von unabhängigen Produkten als ein einziges Produkt lizenziert. Diese Pakete werden von den Lieferanten im Lizenzzertifikat (Lizenzfile) definiert und nicht in der Software. So ist es möglich, dass der Hersteller oder Lieferant die einzelnen Komponenten des Paketes bis kurz vor der Auslieferung anders zusammenstellen kann. Dadurch kann auf kurzfristige Bedürfnisse und Umstellungen des Kunden reagiert werden.

Package Suites

Bei der Lizenzierungsart Package Suites werden dem Anwender Pakete mit einzelnen Paketkomponenten verkauft. Diese Pakete unterliegen aber der Restriktion, dass die Paketkomponenten einer einzelnen Lizenz nicht gleichzeitig genutzt werden dürfen.

Upgrade Versionsdatum

Durch diese Lizenzierungsart bekommt der Kunde, also der Lizenznehmer das Recht, Versionen einer Software zu nutzen, die an oder vor dem Versionsdatum erstellt wurden. Das Recht für Upgrades wird also durch ein Versionsdatum definiert. Dadurch müssen dem Kunden keine neuen Lizenzen ausgestellt werden, wenn er zuvor durch diese Lizenzierungsart spezifische Upgrade-Rechte bis zu einem festgelegten Datum erworben hat.

Upgrade Versionsnummer

Bei Upgrade Versionsnummer wird das Recht für Upgrades durch eine Versionsnummer und nicht durch das Versionsdatum definiert. So erhält der Kunde das Recht, Versionen, die gleich oder kleiner sind als die im Lizenzfile definierte Version, zu nutzen. Wenn der Kunde also spezifische Upgrade-Rechte bis zu einer spezifischen Version erworben hat, müssen ihm demnach keine neuen Lizenzen bis zu dieser Version ausgestellt werden.

Kapazitäts-Lizenzierung

Da Computer mit hoher Performance mehr gleichzeitige Lizenzen zur Nutzung benötigen als solche mit niedriger Performance, wird die Kapazitäts-Lizenzierung für Anwendungen mit intensivem Rechenaufwand verwendet. So kann die potenziell größere Softwarenutzung auf einem Rechner mit hoher Performance kompensiert werden.

Domänen-Lizenz (Company-Lizenz)

Die Lizenz läuft auf jedem Computer einer spezifischen Internet-Domäne, z.B. company.com. Domänen-Lizenzen werden für unternehmensweite Lizenzierung verwendet.

Gruppierung nach Nutzer, Rechner oder Bildschirm

Bei der Lizenzierungsart „Gruppierung nach Nutzern, Rechner oder Bildschirm“ werden Regeln definiert, nach welchen auch die mehr als einfache Nutzung einer Applikation als nur eine einzige Nutzung zählt. Dies ist beispielsweise hilfreich, wenn ein Nutzer

dieselbe Anwendung auf mehreren Computern gleichzeitig startet. In diesem Fall kann die Nutzung als eine einzige Nutzung oder auch Mehrfachnutzung einer Lizenz zählen, wenn ein entsprechender Lizenzierungsvertrag abgeschlossen wurde.

Floating (gleichzeitige) für Rechnerliste

Beim „Floating für Rechnerliste“ können Lizenzen in einem Netzwerk gleichzeitig genutzt werden. Dabei müssen aber die zugelassenen Rechner in einer speziellen Rechnerliste definiert werden.

Post-use-payment

Wie der Name schon anzeigt, werden beim „Post-use-payment“ die Gebühren erst nachträglich gezahlt. So wird beispielsweise die Maximalanzahl gleichzeitiger Benutzter in einem zurückliegenden, genau definierten Zeitabschnitt, so zum Beispiel ein Monat, aufgezeichnet. Diese Aufzeichnungen dienen dann als Basis für die zu zahlenden Gebühren.

Lizenz-Verbleib (linger)

Nach Ablauf eines Lizenzvertrages besteht bei dem Lizenz-Verbleib die Möglichkeit, dass der Nutzer die Lizenz noch weiterhin behält. Die Zeitdauer dieser zusätzlichen Nutzung ist individuell festgelegt.

Überziehen (overdraft)

Hier wird dem Kunden gestattet, n mehr Lizenzen zu nutzen, als er eigentlich erworben hat. So müssen dem Kunden entweder momentane Spitzennutzungen der jeweiligen Lizenzen als Extralizenz in Rechnung gestellt werden, oder der Kunde zahlt in Verbindung mit einem „pay-per-use“-Lizenz-Modell. Bei dieser Nutzung kann der Kunde den Überziehungsumfang eingrenzen, um so die Anwendung des „pay-per-use“-Modells besser kontrollieren zu können.

Pay-per-use

Bei „pay-per-use“ orientieren sich die Lizenzgebühren an der tatsächlichen Nutzung. Es wird also die Nutzungshäufigkeit festgestellt und anhand dieser werden die Lizenzgebühren berechnet. Die Feststellung der Nutzungshäufigkeit kann dabei durch Computermetrik erfolgen (CPU-Nutzung, Uhrzeiten, usw.) oder durch an der Art des Anwenderprogramms ausgerichteten Metriken.

Netzwerk-Segmente (site licence)

Netzwerk-Segmente können definiert werden durch eine Liste von IP-Adressen, so genannten „wild cards“. Dadurch kann gewährleistet werden, dass bestimmte Lizenzen auf gewisse Regionen, Abteilungen oder Unternehmensbereiche begrenzt werden und ausschließlich dort nutzbar sind.

Ersetzen früherer Lizenzen (supersede)

Die Lizenzen, die mit dieser Lizenzierungsart eingesetzt werden, haben ein explizit oder implizit gesetztes Startdatum. Die Bedingung dieser Lizenzierungsart ist, dass alle früheren Lizenzen ihre Gültigkeit verlieren. So kann gewährleistet werden, dass früher ausgestellte Lizenzen zurückgezogen werden.

10.2. Wartungsverträge

Maintenance

Die Berechtigung für Inhaber von Software-Lizenzen, innerhalb der Laufzeit des Lizenzvertrages die jeweils aktuellste Version zu installieren. Der Kunde hat also Anspruch auf alle Verbesserungen, Fehlerbehebungen, Aktualisierungen und Upgrades und wird automatisch über verfügbare Upgrades informiert.

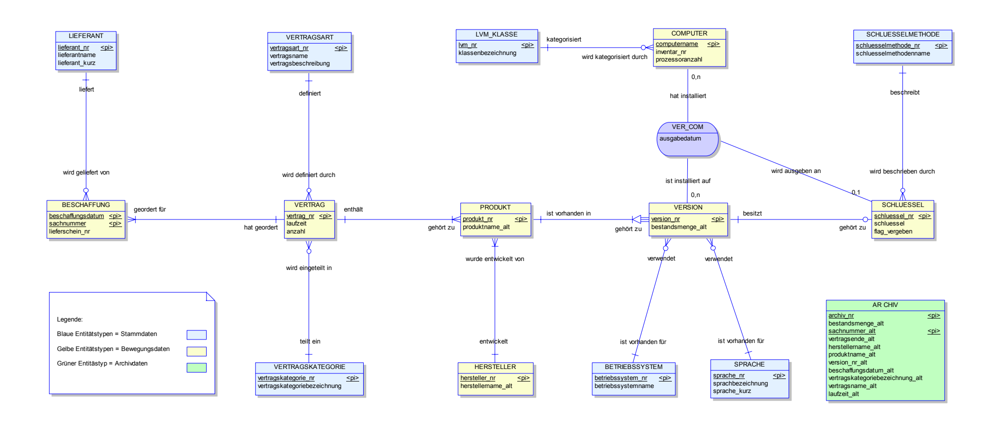
Renewal

Nach Ablauf einer Lizenz bieten manche Unternehmen die Option eines Renewals im direkten Anschluss an den vorausgehenden Vertrag. Durch solch einen Renewal-Vertrag ist es dem Anwender möglich, die bereits abgelaufene Lizenz zu erneuern und diese für einen bestimmten Zeitraum wieder zur Verfügung gestellt zu bekommen.

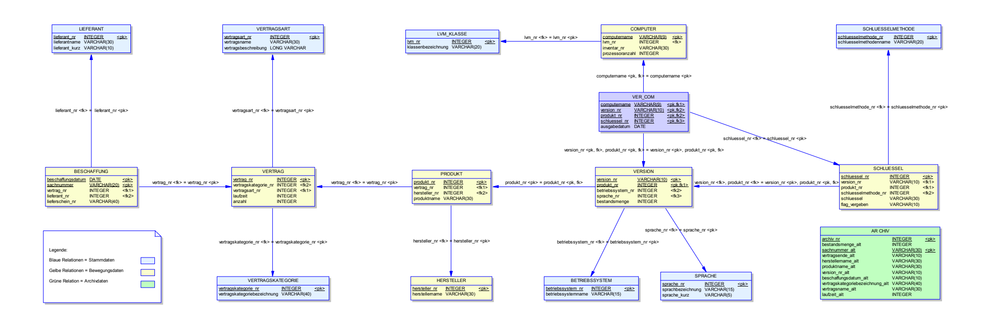
Update

Ein Update ist eine aktualisierte Version einer Software oder Hardware, die eine ältere Version des gleichen Produkts ersetzt. Üblicherweise verkaufen Softwareunternehmen Updates zu einem günstigeren Preis. Meistens muss man nachweisen, dass man im Besitz einer älteren Version (Vollversion) ist, um Rabatt zu erhalten.

10.3. *Entity-Relationship-Modell des License-Management-System*



10.4. *Relationales Modell des License-Management-System*



10.5. *Relationales Modell in SQL*

```
----------------------------------------------------------------
--  Drop Indizes
----------------------------------------------------------------
DROP INDEX prod_bez_idx;
DROP INDEX herst_name_idx;
DROP INDEX c_inventar_nr_idx;
DROP INDEX vart_name_idx;
DROP INDEX vkat_bez_idx;
DROP INDEX lschein_nr_idx;
DROP INDEX lieferant_bez_idx;

----------------------------------------------------------------
--  Drop Sequences
----------------------------------------------------------------
DROP SEQUENCE seq_lvm_klasse;
DROP SEQUENCE seq_schluesselmethode;
DROP SEQUENCE seq_betriebssystem;
DROP SEQUENCE seq_sprache;
DROP SEQUENCE seq_vertragsart;
DROP SEQUENCE seq_vertrag;
DROP SEQUENCE seq_vertragskategorie;
DROP SEQUENCE seq_lieferant;
DROP SEQUENCE seq_hersteller;
DROP SEQUENCE seq_produkt;
DROP SEQUENCE seq_schluessel;
DROP SEQUENCE seq_archiv;

----------------------------------------------------------------
--  Drop Tables
----------------------------------------------------------------
DROP TABLE BETRIEBSSYSTEM;
DROP TABLE SPRACHE;
DROP TABLE SCHLUESSEL;
DROP TABLE SCHLUESSELMETHODE;
DROP TABLE LIEFERANT;
DROP TABLE BESCHAFFUNG;
DROP TABLE VERTRAGSKATEGORIE;
DROP TABLE VERTRAGSART;
DROP TABLE VERTRAG;
DROP TABLE VER_COM;
DROP TABLE LVM_KLASSE;
DROP TABLE COMPUTER;
DROP TABLE VERSION;
DROP TABLE HERSTELLER;
DROP TABLE PRODUKT;
DROP TABLE ARCHIV;
```

```
----------------------------------------------------------------
--  Tables and Primary Keys
----------------------------------------------------------------

CREATE TABLE PRODUKT (
    produkt_nr INTEGER NOT NULL,
    produktname VARCHAR(30),
    sub_produkt_nr INTEGER,
    hersteller_nr INTEGER NOT NULL,
    vertrag_nr INTEGER,
    PRIMARY KEY (produkt_nr)
);

CREATE TABLE HERSTELLER (
    hersteller_nr INTEGER NOT NULL,
    herstellername VARCHAR(30),
    PRIMARY KEY (hersteller_nr)
);

CREATE TABLE VERSION (
    version_nr VARCHAR(10) NOT NULL,
    produkt_nr INTEGER NOT NULL,
    betriebssystem_nr INTEGER,
    sprache_nr INTEGER,
    bestandsmenge INTEGER,
    vertrag_nr INTEGER,
    PRIMARY KEY (version_nr, produkt_nr)
);

CREATE TABLE COMPUTER (
    computername VARCHAR(9) NOT NULL,
    inventar_nr varchar(30),
    lvm_nr INTEGER,
    prozessoranzahl INTEGER,
    PRIMARY KEY (computername)
);

CREATE TABLE LVM_KLASSE (
    lvm_nr INTEGER NOT NULL,
    klassenbezeichnung VARCHAR(20),
    PRIMARY KEY (lvm_nr)
);

CREATE TABLE VER_COM (
    computername VARCHAR(9) NOT NULL,
    version_nr VARCHAR(10) NOT NULL,
    produkt_nr INTEGER NOT NULL,
    ausgabedatum DATE,
    schluessel_nr INTEGER DEFAULT 0,
    PRIMARY KEY (computername, version_nr, produkt_nr)
);
```

```
CREATE TABLE VERTRAG (
    vertrag_nr INTEGER NOT NULL,
    sub_vertrag_nr INTEGER,
    laufzeit INTEGER DEFAULT 0,
    anzahl INTEGER,
    einzelpreis DECIMAL(8,2),
    vertragsart_nr INTEGER,
    beschaffungsdatum DATE,
    sachnummer VARCHAR(20),
    vertragskategorie_nr INTEGER,
    PRIMARY KEY (vertrag_nr)
);

CREATE TABLE VERTRAGSART (
    vertragsart_nr INTEGER NOT NULL,
    vertragsname VARCHAR(30),
    vertragsbeschreibung VARCHAR(1000) DEFAULT NULL,
    PRIMARY KEY (vertragsart_nr)
);

CREATE TABLE VERTRAGSKATEGORIE (
    vertragskategorie_nr INTEGER NOT NULL,
    vertragskategoriebezeichnung VARCHAR(40),
    PRIMARY KEY (vertragskategorie_nr)
);

CREATE TABLE BESCHAFFUNG (
    beschaffungsdatum DATE NOT NULL,
    sachnummer VARCHAR(20) NOT NULL,
    lieferschein_nr VARCHAR(40),
    lieferant_nr INTEGER NOT NULL,
    PRIMARY KEY (beschaffungsdatum, sachnummer)
);

CREATE TABLE LIEFERANT (
    lieferant_nr INTEGER NOT NULL,
    lieferantname VARCHAR(30),
    lieferant_kurz VARCHAR(10),
    PRIMARY KEY (lieferant_nr)
);

CREATE TABLE SCHLUESSELMETHODE (
    schluesselmethode_nr INTEGER NOT NULL,
    schluesselmethodenname VARCHAR(20),
    PRIMARY KEY (schluesselmethode_nr)
);

CREATE TABLE SCHLUESSEL (
    schluessel_nr INTEGER NOT NULL,
    schluessel VARCHAR(30),
    schluesselmethode_nr INTEGER,
    version_nr VARCHAR(10),
    produkt_nr INTEGER,
      flag_vergeben VARCHAR(10),
    PRIMARY KEY (schluessel_nr)
);
```

```
CREATE TABLE SPRACHE (
    sprache_nr INTEGER NOT NULL,
    sprachbezeichnung VARCHAR(15),
    sprache_kurz VARCHAR(5),
    PRIMARY KEY (sprache_nr)
);

CREATE TABLE BETRIEBSSYSTEM (
    betriebssystem_nr INTEGER NOT NULL,
    betriebssystemname VARCHAR(15),
    PRIMARY KEY (betriebssystem_nr)
);

CREATE TABLE ARCHIV (
    archiv_nr INTEGER NOT NULL,
    bestandsmenge_alt INTEGER DEFAULT 0,
    sachnummer_alt VARCHAR(30) NOT NULL,
    vertragsende_alt VARCHAR(10),
    herstellername_alt VARCHAR(30) DEFAULT NULL,
    produktname_alt VARCHAR(30) DEFAULT NULL,
    version_nr_alt VARCHAR(10) DEFAULT NULL,
    beschaffungsdatum_alt VARCHAR(10),
    vertragskategoriebezeichnung_alt VARCHAR(40) DEFAULT NULL,
    vertragsname_alt VARCHAR(30) DEFAULT NULL,
    laufzeit_alt INTEGER DEFAULT 0,
    PRIMARY KEY (archiv_nr, sachnummer_alt)
);




----------------------------------------------------------------
--  Foreign Keys
----------------------------------------------------------------

ALTER TABLE PRODUKT
    ADD FOREIGN KEY (hersteller_nr)
    REFERENCES HERSTELLER (hersteller_nr)
    ON DELETE NO ACTION;

ALTER TABLE PRODUKT
    ADD FOREIGN KEY (vertrag_nr)
    REFERENCES VERTRAG (vertrag_nr)
    ON DELETE CASCADE;

ALTER TABLE VERSION
    ADD FOREIGN KEY (produkt_nr)
    REFERENCES PRODUKT (produkt_nr)
    ON DELETE CASCADE;


ALTER TABLE version
    ADD FOREIGN KEY (betriebssystem_nr)
    REFERENCES BETRIEBSSYSTEM (betriebssystem_nr)
    ON DELETE NO ACTION;
```

```
ALTER TABLE version
    ADD FOREIGN KEY (sprache_nr)
    REFERENCES SPRACHE (sprache_nr)
    ON DELETE NO ACTION;

ALTER TABLE VERSION
    ADD FOREIGN KEY (vertrag_nr)
    REFERENCES VERTRAG (vertrag_nr);

ALTER TABLE COMPUTER
    ADD FOREIGN KEY (lvm_nr)
    REFERENCES LVM_KLASSE (lvm_nr)
    ON DELETE NO ACTION;

ALTER TABLE VER_COM
    ADD FOREIGN KEY (computername)
    REFERENCES COMPUTER (computername)
    ON DELETE CASCADE;

ALTER TABLE VER_COM
    ADD FOREIGN KEY (version_nr,produkt_nr)
    REFERENCES VERSION (version_nr, produkt_nr);

ALTER TABLE VER_COM
    ADD FOREIGN KEY (schluessel_nr)
    REFERENCES SCHLUESSEL (schluessel_nr);

ALTER TABLE VERTRAG
    ADD FOREIGN KEY (vertragsart_nr)
    REFERENCES VERTRAGSART (vertragsart_nr)
    ON DELETE NO ACTION;

ALTER TABLE VERTRAG
    ADD FOREIGN KEY (vertragskategorie_nr)
    REFERENCES VERTRAGSKATEGORIE (vertragskategorie_nr)
    ON DELETE NO ACTION;

ALTER TABLE BESCHAFFUNG
    ADD FOREIGN KEY (lieferant_nr)
    REFERENCES LIEFERANT (lieferant_nr)
    ON DELETE NO ACTION;

ALTER TABLE BESCHAFFUNG
    ADD FOREIGN KEY (vertrag_nr)
    REFERENCES BESCHAFFUNG (vertrag_nr)
    ON DELETE CASCADE;

ALTER TABLE SCHLUESSEL
    ADD FOREIGN KEY (schluesselmethode_nr)
    REFERENCES SCHLUESSELMETHODE (schluesselmethode_nr)
    ON DELETE NO ACTION;

ALTER TABLE SCHLUESSEL
    ADD FOREIGN KEY (version_nr,produkt_nr)
    REFERENCES VERSION (version_nr, produkt_nr)
    ON DELETE CASCADE;
```

```
-----------------------------------------------------------------
--  CREATE Sequences
-----------------------------------------------------------------

CREATE SEQUENCE  seq_lvm_klasse
START WITH 0
INCREMENT BY 1
MAXVALUE 100
NO MINVALUE
NO CYCLE
NO CACHE;

CREATE SEQUENCE  seq_schluesselmethode
START WITH 0
INCREMENT BY 1
MAXVALUE 100
NO MINVALUE
NO CYCLE
NO CACHE;

CREATE SEQUENCE  seq_betriebssystem
START WITH 0
INCREMENT BY 1
MAXVALUE 100
NO MINVALUE
NO CYCLE
NO CACHE;

CREATE SEQUENCE  seq_sprache
START WITH 0
INCREMENT BY 1
MAXVALUE 100
NO MINVALUE
NO CYCLE
NO CACHE;

CREATE SEQUENCE  seq_vertragsart
START WITH 0
INCREMENT BY 1
MAXVALUE 10000
NO MINVALUE
NO CYCLE
NO CACHE;



CREATE SEQUENCE  seq_vertrag
START WITH 0
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
NO CYCLE
CACHE 20;

CREATE SEQUENCE  seq_vertragskategorie
START WITH 0
```

```
INCREMENT BY 1
MAXVALUE 100
NO MINVALUE
NO CYCLE
NO CACHE;

CREATE SEQUENCE  seq_lieferant
START WITH 0
INCREMENT BY 1
MAXVALUE 100
NO MINVALUE
NO CYCLE
NO CACHE;

CREATE SEQUENCE  seq_hersteller
START WITH 0
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
NO CYCLE
CACHE 20;

CREATE SEQUENCE  seq_produkt
START WITH 0
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
NO CYCLE
CACHE 20;

CREATE SEQUENCE  seq_schluessel
START WITH 0
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
NO CYCLE
NO CACHE;


CREATE SEQUENCE  seq_archiv
START WITH 0
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
NO CYCLE
NO CACHE;

----------------------------------------------------------------
--  Indizes
----------------------------------------------------------------

CREATE INDEX prod_bez_idx
ON produkt (produktname DESC);

CREATE INDEX herst_name_idx
ON hersteller (herstellername DESC);
```

```
CREATE INDEX c_inventar_nr_idx
ON computer (inventar_nr DESC);

CREATE INDEX vart_name_idx
ON vertragsart (vertragsart_nr DESC);

CREATE INDEX vkat_bez_idx
ON vertragskategorie (vertragskategoriebezeichnung DESC);

CREATE INDEX lschein_nr_idx
ON beschaffung (lieferschein_nr DESC);

CREATE INDEX lieferant_bez_idx
ON lieferant (lieferantname DESC);
```

10.6. *Anwenderhandbuch des License-Management-System*

Anwenderhandbuch



License

Management

System

Stand: 09.10.2004
Version 1.0

1. Einleitung

Das vorliegende Handbuch über das License-Management-System bietet dem Endanwender eine Hilfe bei der Nutzung des entwickelten Programms sein und einen Überblick geben über die Möglichkeiten der Nutzung, die ihm mit diesem System zur Verfügung stehen. Es ermöglicht die einfache und problemlose Anwendung eines Programms, das in erster Linie der Verwaltung und Pflege von Lizenzen dient. Ziel ist eine genaue und schnelle Übersicht über die vorhandenen Lizenzen, was im „Vertragspool" gegeben ist, in dem auch durch ein Warnsystem auf eine drohende Überschreitung des Bestands und vor der illegalen Nutzung von Produkten gewarnt wird.

Aber auch eine Auflistung der nicht mehr genutzten Lizenzen und Produkte kann bei zukünftigen Vertragsverhandlungen ausschlaggebend sein. So bietet das „Archiv" einen Überblick über ehemals genutzte Produkte und Lizenzen, die bereits abgelaufen sind.
Neue Verträge können mit ihren einzelnen Produkten bei „Vertrag hinzufügen" in den Bestand aufgenommen werden, aber auch wieder bei „Vertrag entfernen" aus dem Bestand in das Archiv überführt werden. Ebenso ist es bei Produkten, die zu spezifischen Computern hinzugefügt oder zurückgenommen werden und Computer, die neu in den Bestand mit aufgenommen oder aus diesem entfernt werden.
Bei den Stammdaten stellt es eine Instandhaltungsfunktion zur Verfügung die es dem Anwender ermöglicht, durch Stammdatenpflege immer die aktuellsten Daten zur Verfügung zu haben und so eine problemlose Handhabung des Programms zu gewährleisten.
Letztlich ergeben sich aus den Funktionen, die dieses License-Management-System erfüllen kann, verschiedene Bearbeitungsvorgänge, die sich auf die Computer, die Verträge, die Produkte, die Stammdaten und das Archiv beziehen und im Folgenden einzeln beschrieben werden.

2. Vertragspool

Die Startseite enthält einen „**Vertragspool**“, der Auskunft über alle vorhandenen Verträgen mit ihren jeweiligen Produkten gibt. Ein Vertrag definiert dabei eine Software-Lizenz, eine Wartungslizenz oder eine Vollversion.

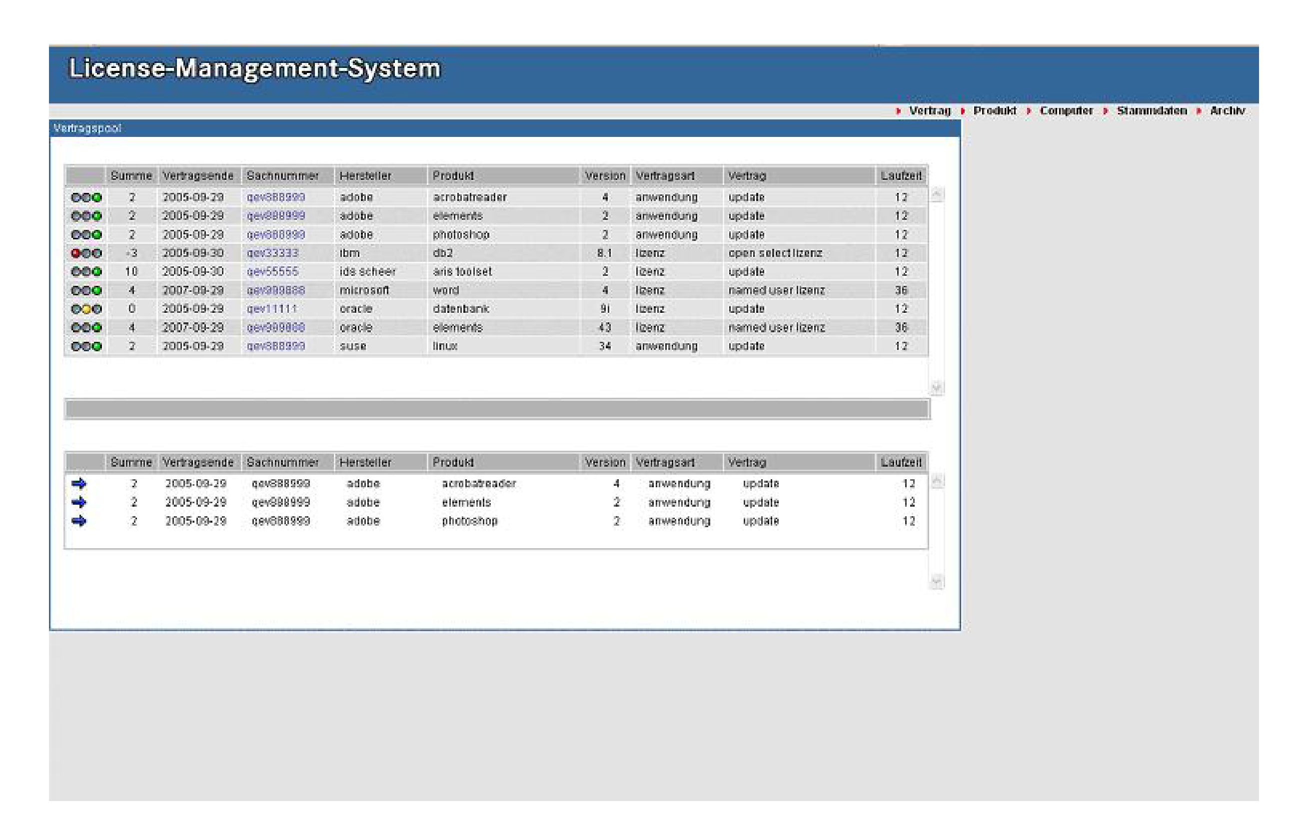


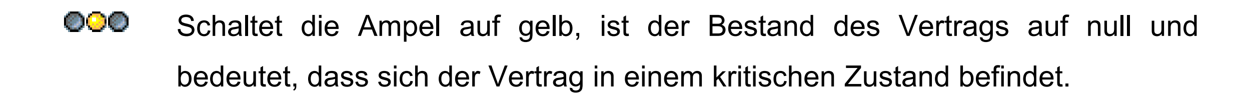
Abbildung: Darstellung des Vertragspools mit dem zur Verfügung stehenden aktiven Vertragsbestand

In diesem Pool werden alle Lizenzen aufgeführt und durch bestimmte Attribute genau definiert, wobei die „*Laufzeit* “ immer in Monaten berechnet wird.

<u>Die Funktion des *Ampelsystems*:</u>

Das Ampelsystem, das vor den jeweiligen Vertrag gestellt ist, zeigt an, ob das Vertragsende bald erreicht ist und somit ein neuer Vertrag beziehungsweise eine Verlängerung des alten Vertrags vorgenommen werden sollte.

Ist die Ampel grün, bedeutet dies, dass die Menge der zur Verfügung stehenden Lizenzen des Vertrags größer oder gleich null ist.



Schaltet die Ampel auf gelb, ist der Bestand des Vertrags auf null und bedeutet, dass sich der Vertrag in einem kritischen Zustand befindet.

Springt die Ampel auf rot, bedeutet dies, dass der Vertrag die gekaufte Bestandsmenge unterschritten hat.

Durch diese Ampelfunktionalität soll die Rechtmäßigkeit der Lizenznutzung gewährleistet werden und an den rechtzeitigen Nachkauf von Lizenzen erinnert werden.

Die Funktion der zweiten Tabelle:

Wenn zu einem Vertrag *mehrere Produkte* gehören, können diese in einer weiteren Tabelle angezeigt werden, die sich ebenfalls auf der Seite „**Vertragspool**" befindet. Diese Tabelle ist anfangs leer und wird nur gefüllt, wenn ein spezieller Vertrag aus der ersten Tabelle angeklickt wird. Enthält dieser Vertrag mehrere Produkte, werden diese mit ihren jeweiligen Attributen in der unteren Tabelle angezeigt.

	Summe	Vertragsende	Sachnummer	Hersteller	Produkt	Version	Vertragsart	Vertrag	Laufzeit
	2	2005-09-29	qev888999	adobe	acrobatreader	4	anwendung	update	12
	2	2005-09-29	qev888999	adobe	elements	2	anwendung	update	12
	2	2005-09-29	qev888999	adobe	photoshop	2	anwendung	update	12
	2	2005-09-29	qev888999	suse	linux	34	anwendung	update	12

Abbildung: Darstellung der zu einem Vertrag dazugehörigen Produkte

Weiterführende Links auf dieser Seite:

Der Vertragspool, wie auch alle anderen Seiten des License-Management-System bietet mithilfe der Navigationsstruktur mehrere Links zu weiteren Seiten an, auf denen eine tiefere und weitergehende Bearbeitung der Vertragsdaten angeboten wird.



Die Links bieten eine Weiterleitung zu „**Vertrag**" an, wobei hier ausgewählt werden kann zwischen „*Vertrag suchen*", „*Vertrag hinzufügen*" und „*Vertrag entfernen*".

Bei dem Link „**Produkt**“ bestehen die Möglichkeiten „*Produkt zuweisen*“, „*Produkt zurücknehmen*“ und „*Produkt suchen*“, ebenso wie bei dem Link „**Computer**“, der auch die Funktionen „*Computer hinzufügen*“, „*Computer entfernen*“ und „*Computer suchen*“ anbietet.

Bei den „**Stammdaten**“ ergibt sich die Möglichkeit, diese zu verwalten und gegebenenfalls zu erweitern oder anderweitig zu verändern.

Das „**Archiv**“ verzeichnet alle Produkte mit ihren Verträgen, die nicht mehr im Datenbestand aktiv sind. Diese Daten werden dort gespeichert, damit beispielsweise bei neuen Vertragsabschlüssen bessere Konditionen erreicht werden können, weil bereits bestimmte Produkte eines Herstellers bezogen worden sind. Auch können bereits im Archiv gespeicherte Daten wieder zurück in den aktiven Datenbestand gebracht werden, wenn entsprechende Verträge abgeschlossen werden.

3. Vertrag

3.1. Vertrag hinzufügen

Die Seite „**Vertrag hinzufügen**“ ermöglicht es, neu abgeschlossene Verträge in den Datenbestand mit aufzunehmen. Gibt es zu einem Vertrag mehrere Produkte, können diese ebenfalls einzeln hinzugefügt werden und garantieren so die Konsistenz des Datenbestandes.

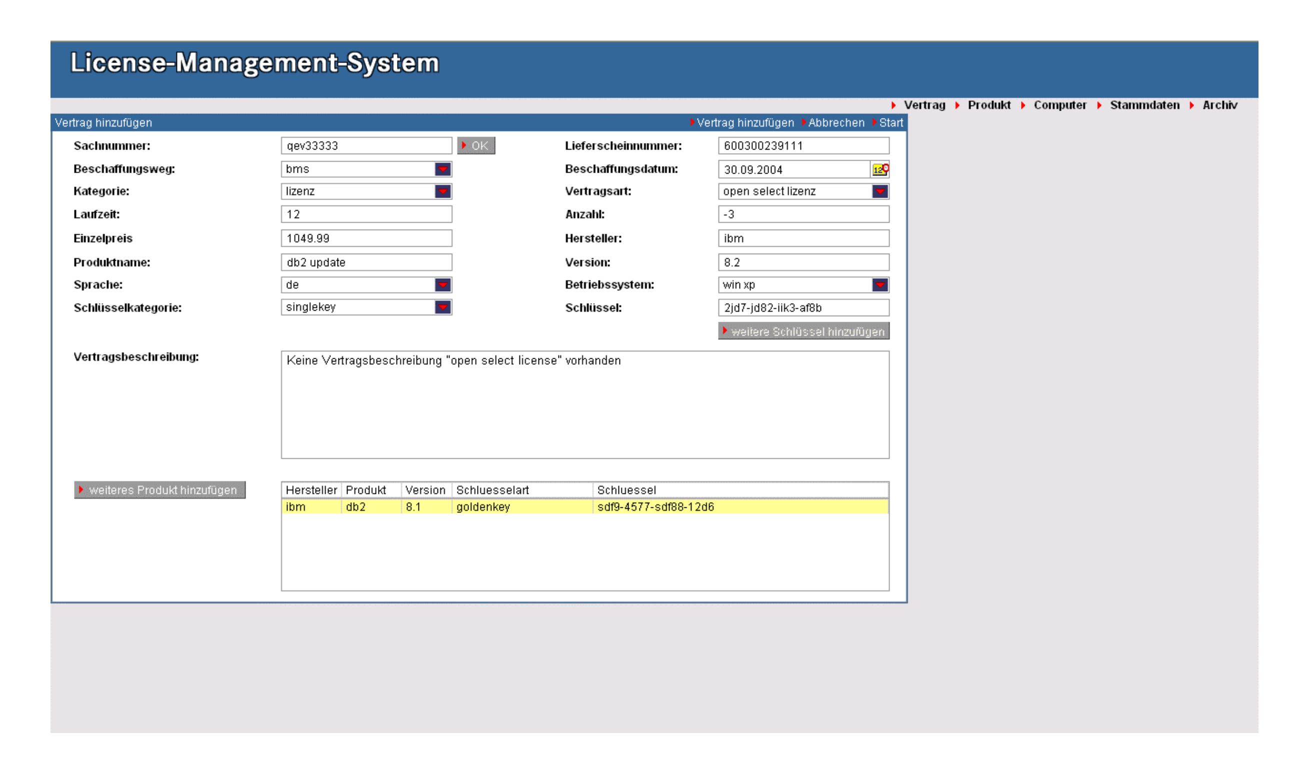


Abbildung: Darstellung der Seite „Vertrag hinzufügen“, mit der es möglich ist, einen Vertrag beziehungsweise zusätzliche Vertragskomponenten zum aktiven Datenbestand hinzuzufügen

Auf dieser Seite werden einige Dropdown-Felder angeboten, bei welchen der Anwender aus einer vorgegebenen Anzahl von Stammdaten auswählen kann, andere Felder müssen manuell ausgefüllt werden, wobei bei dem Feld „*Einzelpreis*“ zu beachten ist, dass die Kommastelle bei der Preiseingabe durch einen Punkt zu tätigen ist.

Bei dem Dropdown-Feld „Beschaffungsweg“ werden die Möglichkeiten „bms“ und „banf“ angeboten. Das Feld „Kategorie“ stellt „anwendung“, „lizenz“ und „wartung“ zur jeweiligen Auswahl.

<u>Die Felder „Schlüsselkategorie“ und „Schlüssel“:</u>

Bei dem Feld „Schlüsselkategorie“ ist mithilfe des Dropdown-Feldes zwischen drei verschiedenen Schlüsseln zu wählen. Diese Wahl hat Auswirkungen auf das Feld „Schlüssel“:

1. Wird der „*concernkey*“ gewählt, kann in dem folgenden Feld „Schlüssel“ keine nähere Angabe gemacht werden, das Feld ist nicht mehr beschreibbar.
2. Bei „*goldenkey*“ wird in das Feld „Schlüssel“ lediglich eine einzige Schlüsselnummer eingetragen. Durch die Nutzung des Buttons „**Vertrag hinzufügen**“ wird, wie bei „concernkey“ auch, der Vertrag gespeichert und auf der Seite des „**Vertragspool**“ angezeigt.
3. Wird jedoch der „*singlekey*“ gewählt, müssen je nach der Anzahl mehrere Schlüssel angegeben werden. Dazu muss nach der Eingabe des ersten Schlüssels ebenfalls der Button „**Vertrag hinzufügen**“ betätigt werden. Dann wird bei einem „singlekey“ automatisch das „Schlüssel“ Feld leer gesetzt und es kann der nächste Schlüssel eingegeben werden. Um diesen zu speichern, ist der Button „**weitere Schlüssel hinzufügen**“ zu nutzen und nach Bestätigung der Frage, ob noch weitere Schlüssel hinzugefügt werden sollen, wird dieses Feld wieder leer gesetzt und kann erneut ausgefüllt werden, bis jedes Produkt seinen individuellen Schlüssel hat.

Weitere Produkte zu diesem Vertrag hinzufügen:

Nach dem Speichern des Vertrags mit seinem Produkt und den jeweiligen Schlüsseln besteht bei allen Schlüsselkategorien die Möglichkeit, weitere Produkte zu diesem Vertrag hinzuzufügen. Dazu muss der Button „**weiteres Produkt hinzufügen**“ gedrückt werden. Danach werden die Produktfelder wieder leer gesetzt, die Vertragsfelder bleiben aber mit den bisherigen Daten ausgefüllt. Die weiteren Produkte werden dann, wie beim ersten Speichervorgang auch, bei Nutzung des Buttons „**Vertrag hinzufügen,**“ gespeichert und auch gleichzeitig auf der Startseite in den Vertragspool eingefügt.

Die Funktion des „**OK**“-Buttons:

1. Bei der erstmaligen Eingabe eines Vertrages müssen alle Felder noch einzeln ausgefüllt werden.
2. Ist der Vertrag aber bereits gespeichert und sollen nachträglich weitere Produkte zu diesem Vertrag hinzugefügt werden, kann die Sachnummer eingegeben werden und die Vertragsfelder werden durch Betätigung des „**OK**“-Buttons automatisch

ausgefüllt. Auf diesen Daten basierend können dann weitere Produktangaben gemacht und das Produkt gespeichert werden.

Dabei gibt es auf dieser, wie auf allen folgenden Seiten, immer die Möglichkeit, durch „**Abbrechen**“ alle Felder null zu setzen und durch „**Start**“ auf die Vertragspool-Seite zu gelangen.

3.2. Vertrag entfernen

Durch die Seite „**Vertrag entfernen**“ können ganze Verträge mit all ihren Produkten entfernt und dadurch in das Archiv gebucht werden und nur einzelne Produkte eines Vertrages entfernt werden. Diese Daten stehen anschließend der aktiven Nutzung des License-Management-System nicht mehr zur Verfügung. So können abgelaufene Produkte oder auch Verträge entfernt und eine illegale Anwendung verhindert werden.

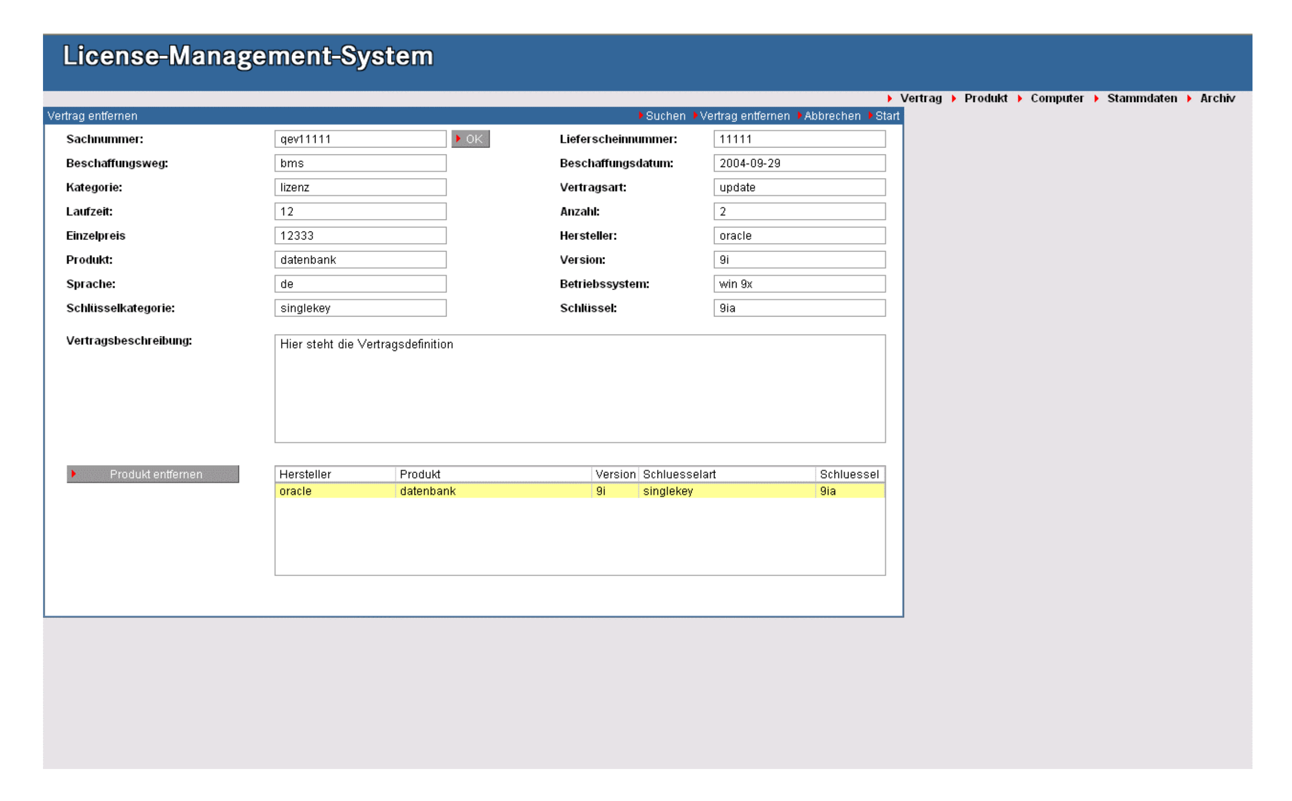


Abbildung: Darstellung der Seite „Vertrag entfernen“. Mit dieser Seite ist es dem Anwender möglich, einen kompletten Vertrag bzw. einzelne Vertragskomponenten aus dem aktiven Datenbestand zu entfernen

Um den genau definierten Vertrag zu finden, der entfernt werden soll, gibt es zwei Möglichkeiten:

1. Die Sachnummer kann eingegeben werden. Durch Betätigung des „**OK**“-Buttons füllen sich die Vertragsfelder automatisch aus und alle zu diesem Vertrag gehörigen Produkte werden aufgeführt.
2. Ist die Sachnummer nicht direkt zur Hand, kann durch den Link „**Vertrag Suchen**“ die passende Seite aufgerufen und hier die Sachnummer ermittelt werden.

Entfernen des gesamten Vertrags:

Soll nun der gesamte Vertrag entfernt werden, wird der Link „**Vertrag entfernen**“ genutzt, wonach der Vertrag in das Archiv überführt wird.

Entfernen einzelner Produkte des Vertrags:

Möchte der Anwender nur ein einzelnes Produkt dieses Vertrags entfernen, muss das spezifische Produkt aus der Liste ausgewählt werden. Dessen Daten werden automatisch in die Produktfelder eingefügt und durch den Button „**Produkt entfernen**“ wird nur dieses Produkt entfernt und kommt in das Archiv.

3.3. Vertrag suchen

Diese Funktion kann, wie bei „Vertrag entfernen," bereits erläutert, genutzt werden, wenn bestimmte Daten zu einem Vertrag gesucht werden.

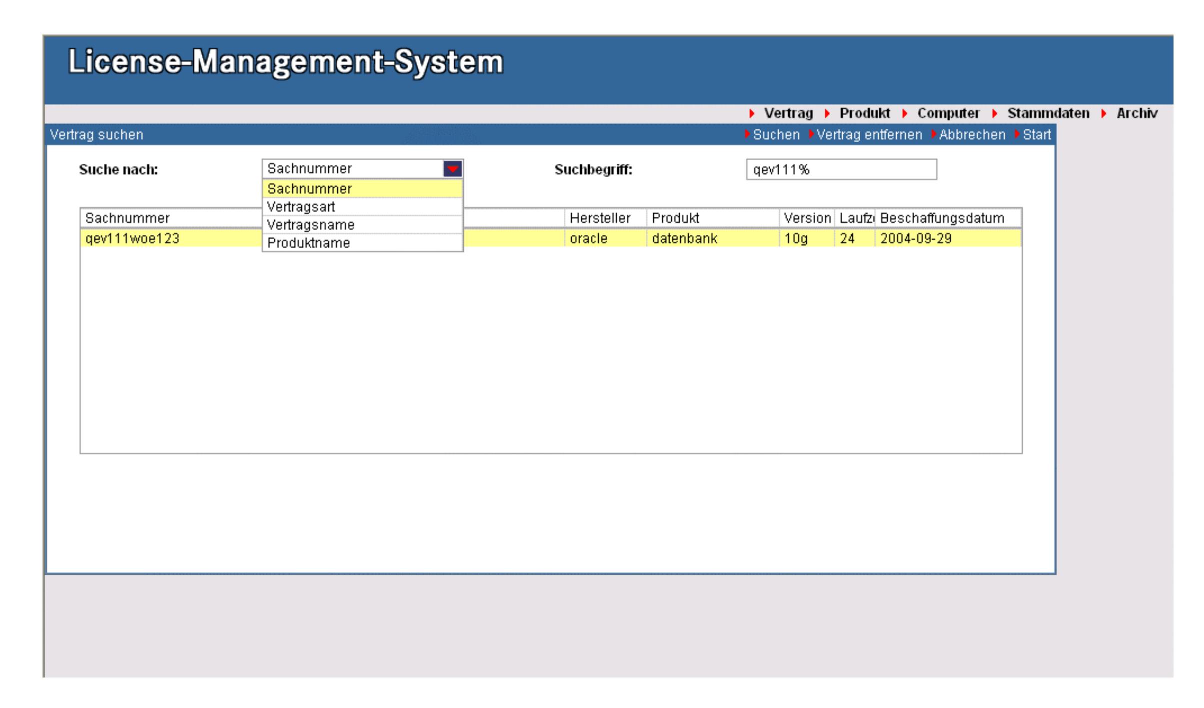


Abbildung: Darstellung der Seite „Vertrag suchen" die es ermöglicht, einen Vertrag aus dem Datenbestand zu suchen

Es werden verschiedene Kriterien in einem Dropdown-Feld angeboten, nach denen gesucht werden kann. So kann im Feld „*Suche nach*" gewählt werden zwischen den Kriterien „Sachnummer", „Vertragsart", „Vertragsname" und „Produktname".
In das Feld „*Suchbegriff*" können passend zum ausgewählten Kriterium entweder ganze Begriffe eingegeben werden oder Teile des Begriffs durch das *Trunkierungszeichen %* ersetzt werden.
Sind diese beiden Felder ausgefüllt, wird durch den Link „**Suchen**" das untere Feld mit allen Verträgen gefüllt, die aus den Angaben ermittelt werden konnten.

Links dieser Seite:

Durch den Link „**Abbrechen**" werden alle Felder wieder null gesetzt und durch den Link „**Start**" gelangt man auf die Vertragspool-Seite.
Wird aber einer der angezeigten Verträge angeklickt und dann auf den Link „**Vertrag entfernen**" gegangen, gelangt man auf die Seite „Vertrag entfernen", auf welcher dann

das Feld der Sachnummer bereits mit den gesuchten Daten ausgefüllt ist und die weiteren Funktionen dieser Seite zur Verfügung stehen.

4. Produkt

4.1. Produkt zuweisen

Die Seite „**Produkt zuweisen**“ ermöglicht es, ein bestimmtes Produkt, das noch frei und nicht vergeben im Vertragspool ist, an einen speziellen Computer zu vergeben. Gleichzeitig ergibt sich hier die Möglichkeit, alle Produkte zu sehen, die auf einem Computer installiert sind.

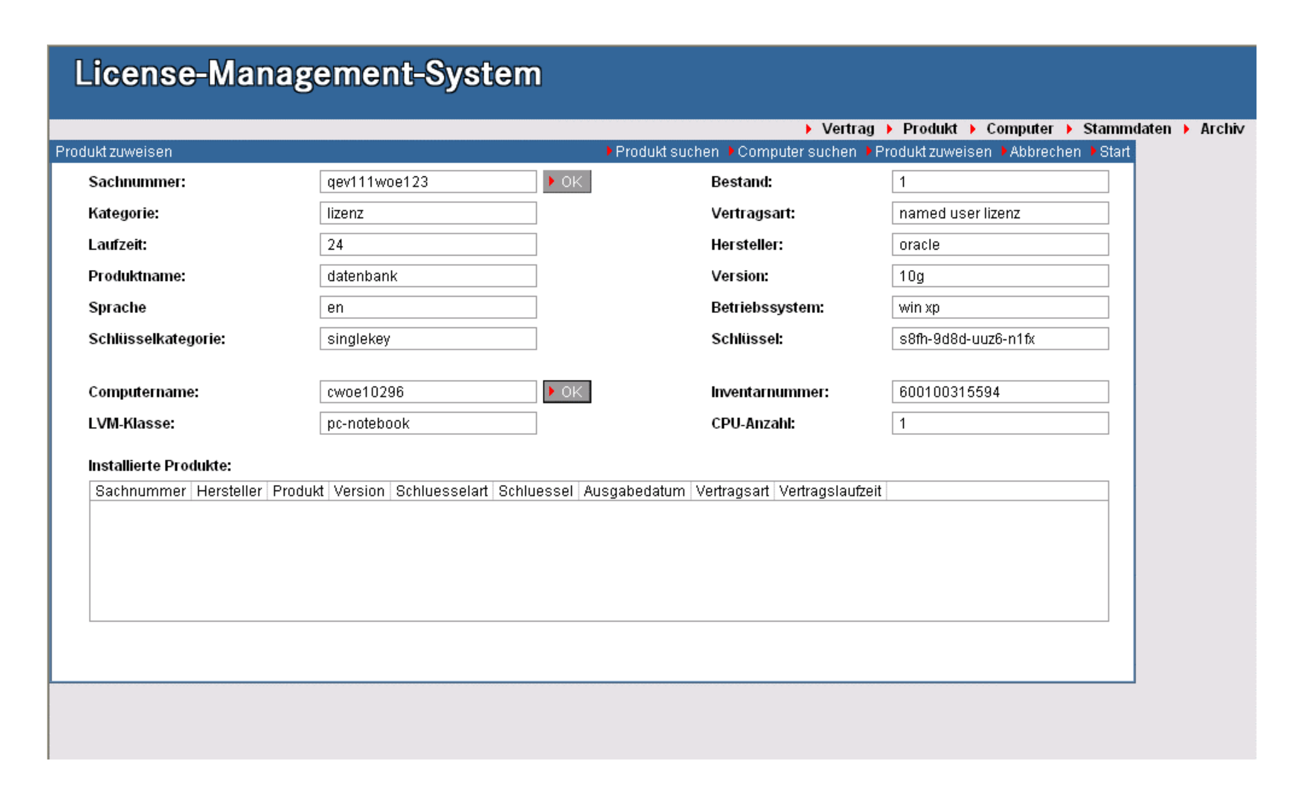


Abbildung: Darstellung der Seite „Produkt suchen“ mit der es möglich ist, ein Produkt auf einen Computer zu dokumentieren

Bei der Suche nach der *Sachnummer* des Produktes, das zugewiesen werden soll, gibt es verschiedene Möglichkeiten.

1. Wenn die Sachnummer eines Produktes bekannt ist und dem Anwender vorliegt, kann diese direkt in das dazugehörige Feld eingegeben werden und die rechtlichen Felder werden durch Nutzung des „**OK**“-Buttons automatisch ausgefüllt.
2. Ist die Sachnummer nicht direkt zur Hand, kann der Link „**Produkt suchen**“ genutzt werden und auf der dazugehörigen Seite nach einem bestimmten Produkt

gesucht werden. Dessen Daten können dann wieder auf die Seite „**Produkt zuweisen**" übergeben werden.

Ebenso ist die Vorgehensweise bei dem Ausfüllen der Computerfelder:

1. Wenn der *Computername* direkt eingegeben werden kann, muss wiederum nur der „**OK**"-Button betätigt werden und die restlichen Felder, die zum Bereich Computer gehören, werden ausgefüllt.
2. Für den Fall, dass der Computername nicht bekannt ist, gibt es den Link zur Seite „**Computer suchen**", auf welcher die Computer ermittelt werden können. Durch Anklicken des gewünschten Computers und den Link „**Produkt zuweisen**" werden die ausgewählten Computerdaten direkt in die Felder eingegeben.

Gleichzeitig werden in der unteren Tabelle alle auf diesem Computer *installierten Produkte* angezeigt und so ergibt sich ein guter Überblick über den Bestand der Produkte auf diesem Computer.

Links dieser Seite:

Sind alle Felder ausgefüllt, kann das ausgewählte Produkt dem definierten Computer durch den Link „**Produkt zuweisen**" übergeben werden und dieses wird in der unteren Tabelle nun ebenfalls als installiertes Produkt angezeigt.
Auch auf dieser Seite können durch „**Abbrechen**" alle Felder auf null gesetzt und durch „**Start**" die Vertragspoolseite aufgerufen werden.

4.2. Produkt zurücknehmen

Ist ein bestimmtes Produkt bereits einem Computer zugewiesen und soll nun aber von diesem zurückgenommen und in den Vertragspool gesetzt werden, tritt die Seite „**Produkt zurücknehmen**" in Funktion.

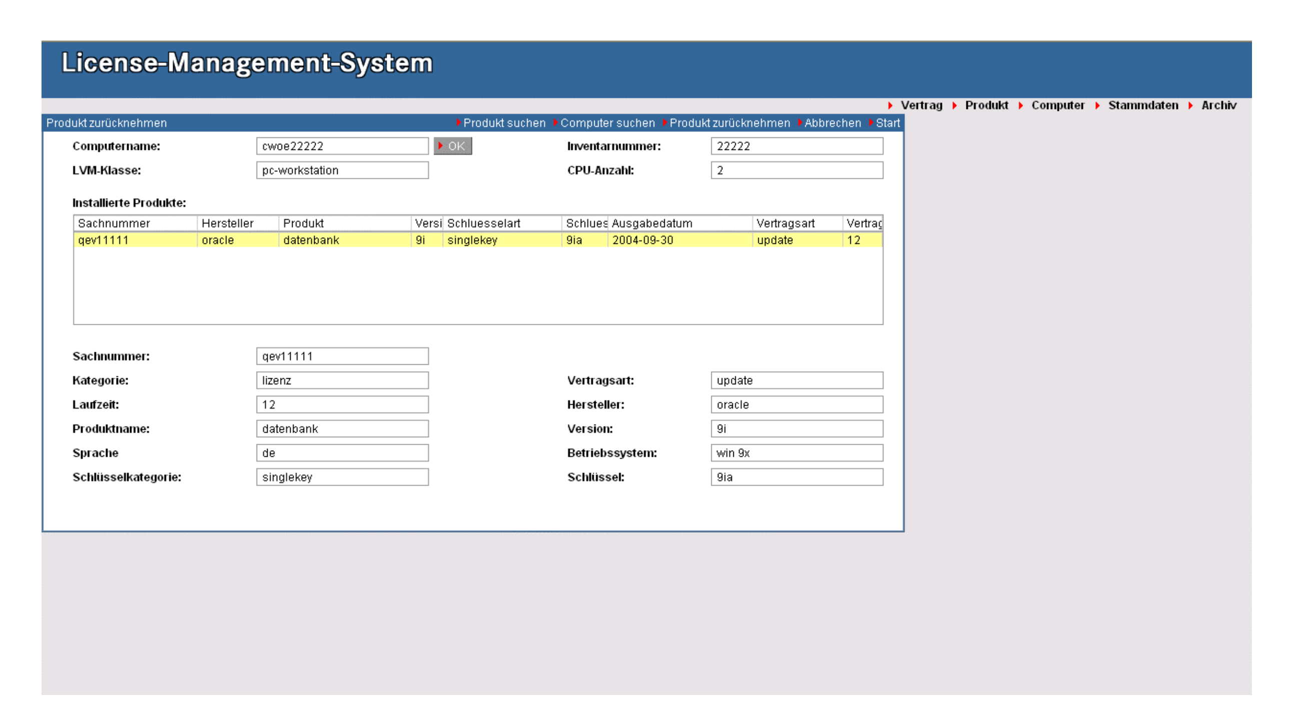


Abbildung: Die Hilfe der Seite „Produkt zurücknehmen“ ist es dem Anwender möglich, ein bereits vergebenes Produkt in den aktiven Datenbestand zurück in den aktiven Datenbestand zu speichern.

Um die Computerfelder auszufüllen, gibt es auch hier wieder mehrere Möglichkeiten:

1. Auch hier wieder ist lediglich der *Computername* einzugeben und die restlichen Computerfelder werden automatisch mithilfe des „**OK**“-Buttons ausgefüllt.
2. Falls der Computername nicht direkt zur Verfügung steht, kann er wieder durch den Link zu „**Computer suchen**“ ermittelt und übergeben werden.

Ist der Computername letztlich eingegeben, werden alle Produkte, die auf diesem Computer installiert sind, angezeigt.

Für die Auswahl des *Produktes*, das zurückgenommen werden soll, gibt es nun ebenfalls zwei Möglichkeiten.

1. Zum einen kann direkt aus der Tabelle mit den installierten Produkten das Gewünschte angeklickt werden, wodurch die Felder zum Produkt ausgefüllt werden.
2. Die Sachnummer des Produktes kann zum anderen auch manuell ausgefüllt werden und die restlichen Felder werden nach dem Drücken des „**OK**“-Buttons ergänzt. Ist die Sachnummer nicht zur Hand, kann diese wiederum durch „**Produkt suchen**“ ermittelt werden.

Sind letztendlich alle Felder ausgefüllt, kann durch den Link „**Produkt zurücknehmen**“ das ausgewählte Produkt von diesem Computer entfernt und zurück in den Vertragspool gegeben werden, wo der Bestand um das zurückgenommene Produkt erhöht wird.

4.3. Produkt suchen

Die bereits erwähnte Funktion „**Produkt suchen**“ ermöglicht das Finden eines bestimmten Produktes, zu welchem man nicht alle nötigen Daten direkt zur Hand hat. Des Weiteren kann man die ermittelten Daten direkt von dieser Seite aus auf andere Seiten übergeben und so ohne Umwege mit diesen weiter arbeiten.

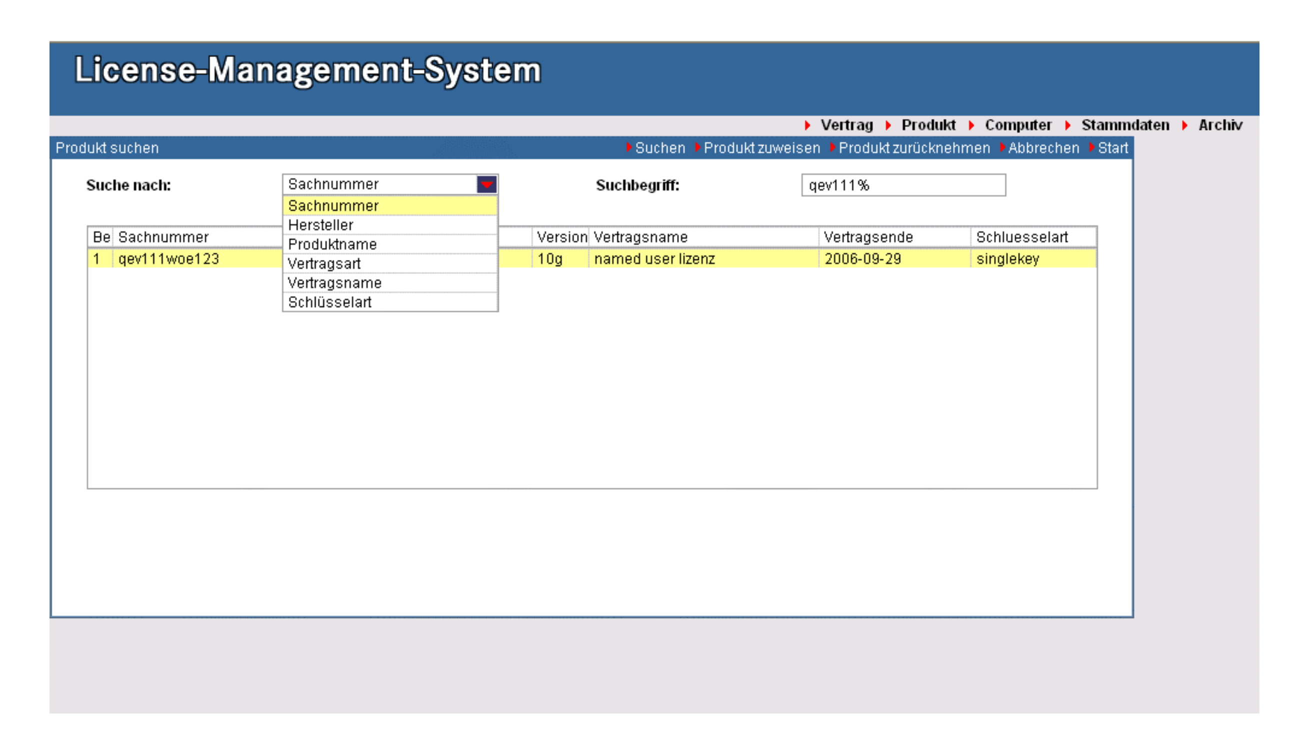


Abbildung: Die oben dargestellte Seite „Produkt suchen“ ermöglicht die Suche nach einem im Datenbestand gespeicherten Produkt

Die Suche erfolgt nach verschiedenen Kriterien, welche aus dem Dropdown-Feld „*Suche nach*“ ausgewählt werden können. Es stehen die Kriterien „Sachnummer“, „Hersteller“, „Produktname“, „Vertragsart“ und „Schlüsselart“ zur Auswahl und zur gewünschten Kategorie kann dann im nebenstehenden Feld ein „*Suchbegriff*“ eingegeben werden, für welches wiederum das *Trunkierungszeichen %* ist.

Links dieser Seite:

Durch „**Suchen**“ werden alle zu diesen Angaben passenden Produkte ausgewählt und in der Tabelle aufgezeigt.

Ist das gewünschte Produkt nicht dabei, können alle Felder durch „**Abbrechen**“ auf null gesetzt werden und die Suche erneut beginnen.

Ist das gewünschte Produkt aber dabei, kann es angeklickt und an eine andere Seite übergeben werden. Hier stehen die Seiten „**Produkt zuweisen**“ und „**Produkt zurücknehmen**“ zur Auswahl.

5. Computer

5.1. Computer hinzufügen

Mithilfe dieser Seite können neue Computer, die bisher nicht im Datenbestand des License-Management-System aufgenommen worden sind, zum Bestand hinzugefügt werden.

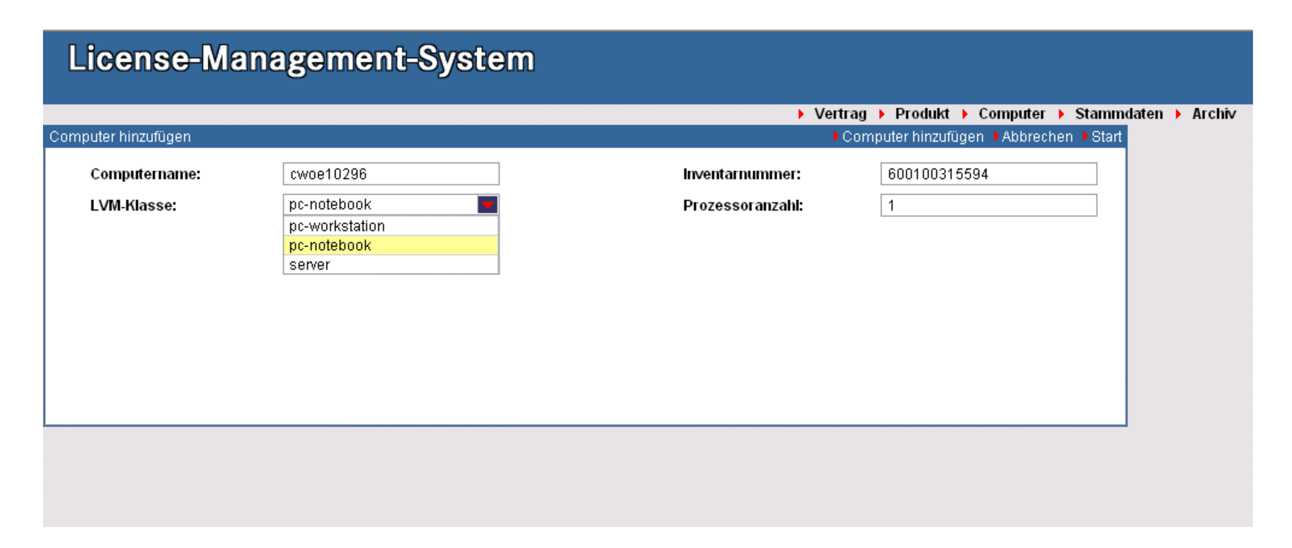


Abbildung: Die Seite „Computer hinzufügen“ ermöglicht es dem Anwender, einen Computer zum Datenbestand des Systems hinzuzufügen

Um einen neuen Computer registrieren zu können, müssen die Felder „Computername“, „Inventarnummer“, „Prozessoranzahl“ und die „LVM-Klasse“ ausgefüllt werden, wobei bei der LVM-Klasse durch ein Dropdown-Feld „PC-Workstation“, „PC-Notebook“ und „Server“ zur Auswahl gegeben sind.

<u>Links dieser Seite:</u>

Sind alle Felder ausgefüllt, kann der beschriebene Computer durch den Link „**Computer hinzufügen**“ in den Bestand mit aufgenommen werden, durch „**Abbrechen**“ werden alle Felder wieder leer und Eingabefehler und Ähnliches können so ausgeglichen werden.

5.2. Computer entfernen

Die Seite „**Computer entfernen**“ ist nötig, wenn ein bisher genutzter Computer aus dem Bestand entfernt werden soll. Hier ist zu beachten, dass ein Computer nur entfernt werden kann, wenn sich keine dokumentierten Verträge mehr auf ihm befinden. Besitzt der Computer noch dokumentierte Verträge, müssen diese auf der Seite „**Produkt zurücknehmen**“ oder durch den Button „**Produkt entfernen**“ zurück in den Vertragspool gebucht werden.

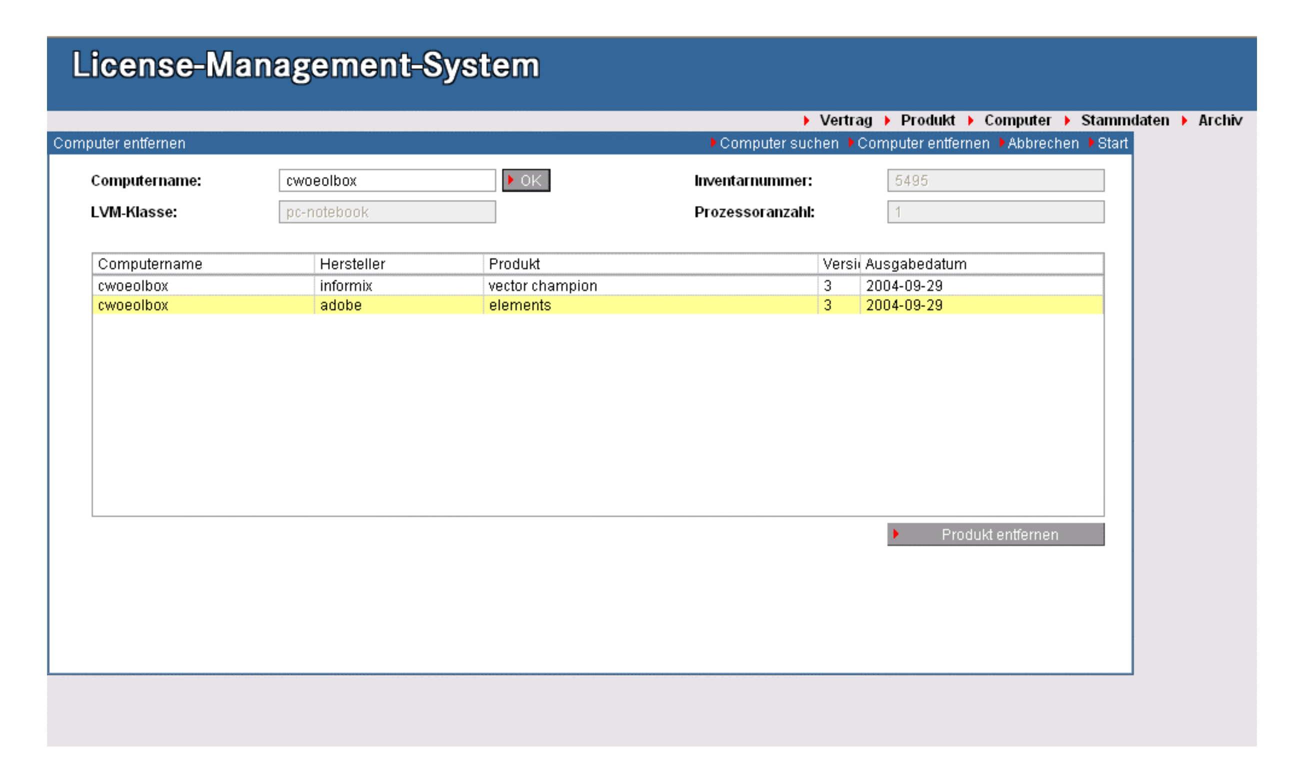


Abbildung: Die Seite „Computer entfernen“ entfernt einen Computer aus dem aktiven Datenbestand des License-Management-Systems

Bei der Eingabe des *Computernamens* ergeben sich zwei Möglichkeiten.

1. Ist hier eine genaue Definition durch den Computernamen möglich, muss nur dieses Feld ausgefüllt und der dazugehörige „**OK**“-Button betätigt werden, wonach alle restlichen Felder ausgefüllt werden.
2. Ist dieser Name nicht zur Hand, steht der Link „**Computer suchen**“ zur Verfügung. Auf dieser Seite können die fehlenden Daten ermittelt und wieder zurück zur Seite „**Computer entfernen**“ übergeben werden.

Sind die Felder ausgefüllt, werden in der Tabelle alle auf diesem Computer *installierten Produkte* angezeigt. Diese Produkte müssen zuerst entfernt werden, bevor der Computer aus dem Datenbestand entfernt werden kann.

Entfernen des Computers:

Sind alle Produkte entfernt, wird der Computer durch den Link „**Computer entfernen**“ aus dem Datenbestand gelöscht.

Entfernen einzelner Produkte des Computers:

Sollen nur einzelne Produkte dieses Computers entfernt werden, der Computer selbst aber nicht, ist die Seite „**Produkt zurücknehmen**“ zu nutzen oder der Button „**Produkt entfernen**“ auf der Seite „**Computer entfernen**“.

5.3. Computer suchen

Wird ein Computer für weitere Bearbeitungsschritte benötigt, aber nicht alle ausschlaggebenden Daten sind vorhanden, kann die Seite „**Computer suchen**“ weiterhelfen.

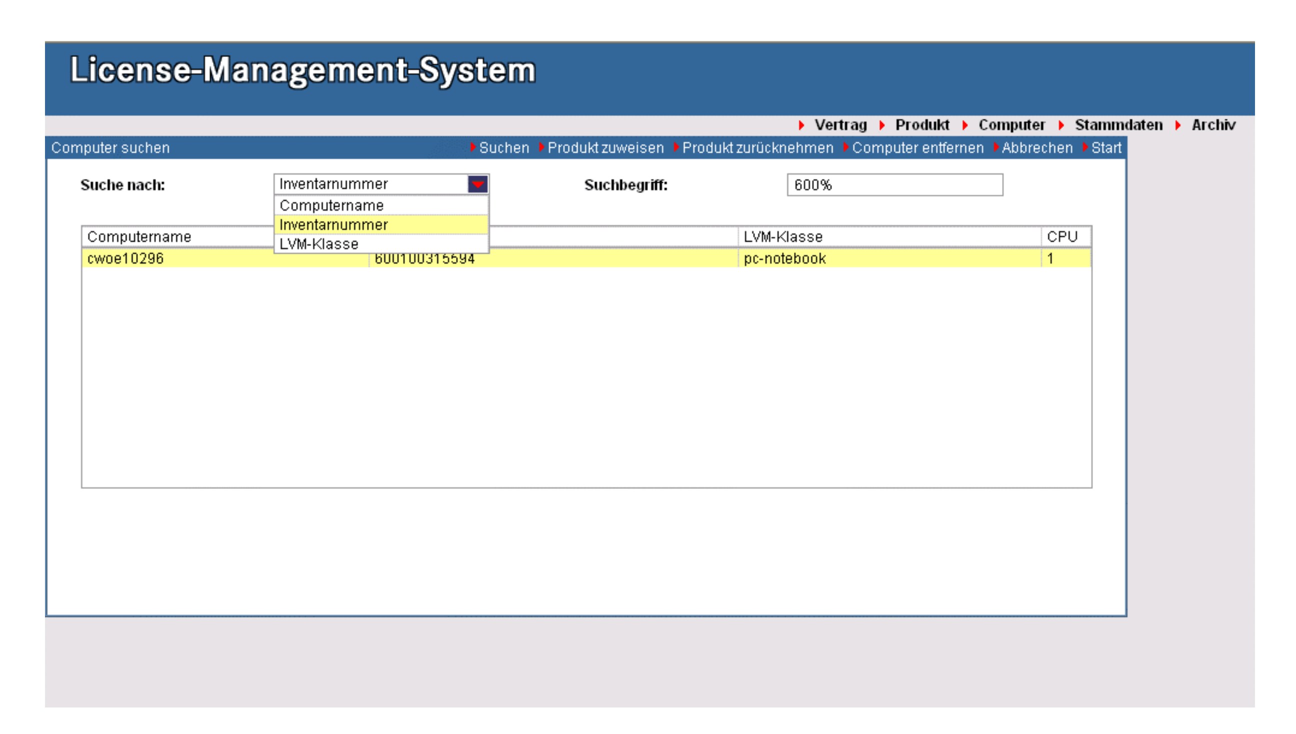


Abbildung: Darstellung der Seite „Computer suchen“ mit der es möglich ist, einen Computer im Datenbestand zu suchen

Hier kann wieder nach verschiedenen Kriterien gesucht werden, welche „Computername“, „Inventarnummer“ und „LVM-Klasse“ umfassen.

Links dieser Seite:

Ist ein Kriterium ausgewählt und ein Suchbegriff, gegebenenfalls mit dem *Trunkierungszeichen %* versehen, eingegeben worden, können durch „**Suchen**“ alle zu diesen Angaben passenden Computer ermittelt und in der Tabelle angezeigt werden. Falls das Gesuchte nicht dabei ist, kann man durch „**Abbrechen**“ wieder alle Felder leer setzten.
Wird der gesuchte Computer aber angezeigt, kann er angeklickt werden und durch die Links „**Produkt zuweisen**“, „**Produkt zurücknehmen**“ und „**Computer entfernen**“ an die jeweilige Seite übergeben werden. Auf diesen Seiten wird das entsprechende Feld „*Computername*“ direkt ausgefüllt und weitere Bearbeitungsschritte können folgen.

6. Stammdaten

Die Stammdatenpflege ist ein sehr wichtiger Bestandteil dieses License-Management-System. Denn durch die gute und gewissenhafte Instandhaltung der dort gespeicherten Daten kann flexibel und individuell auf Veränderungen der Daten reagiert werden.
Der genaue Überblick und so auch schnelle Zugriff auf einzelne Datenbereiche wird dadurch gesichert, dass die Stammdatenpflege in insgesamt 8 verschiedene Seiten unterteilt ist, die jeweils eine eigene Stammdatenkategorie verwalten.
So gibt es eine Seite, welche die „**Vertragsart**“ pflegt.

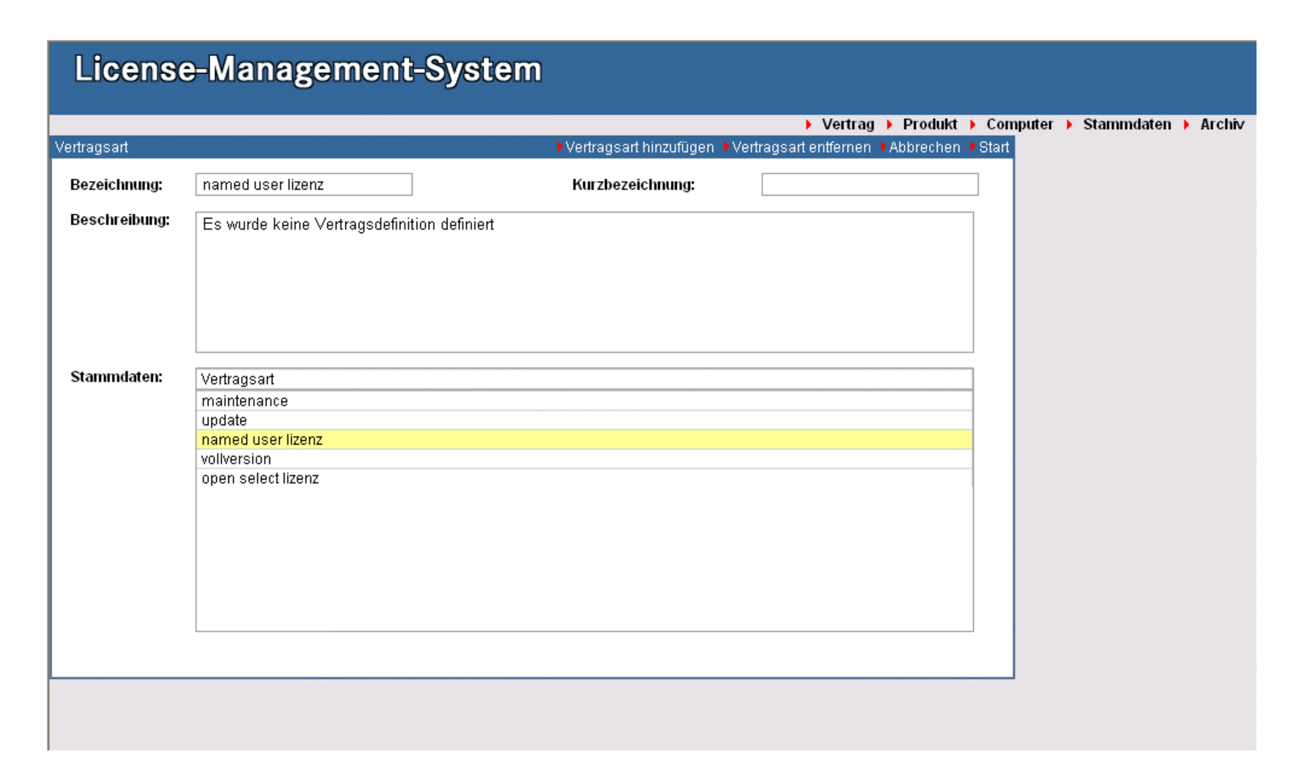


Abbildung: Mithilfe der Seite „Vertragsart“ ist es möglich, Vertragsarten und deren Definitionen als Stammdaten im License-Management-System zu speichern

Hier können in den Feldern „*Bezeichnung*“ und „*Beschreibung*“ neue Vertragsarten mit ihrer genauen Definition eingegeben werden. Durch Betätigung des Links „**Vertrag hinzufügen**“ wird diese neue Vertragsart mit in die Liste der „*Stammdaten*“ aufgenommen.
Soll eine Vertragsart aus dieser Liste und so dem Datenbestand entfernt, werden genügt es, diese in der Liste anzuklicken und die oberen Felder werden automatisch ausgefüllt. Mithilfe des Links „**Vertrag entfernen**“ wird diese spezielle Vertragsart mit ihrer Beschreibung aus der Liste der Stammdaten entfernt. Durch „**Abbrechen**“ sind wiederum alle Felder leer zu setzen und bei Nutzung des Links „**Start**“ gelangt man auf die Vertragspoolseite.

Eine andere Seite pflegt die Stammdaten, die der „**Vertragskategorie**“ zugehören.

License-Management-System

Vertrag Produkt Computer Stammdaten Archiv

Vertragskategorie

Vertragsart hinzufügen Vertragsart entfernen Abbrechen Start

Bezeichnung: wartung

Stammdaten: Vertragskategorie

anwendung

lizenz

wartung

Abbildung: Darstellung der Stammdatenseite „Vertragskategorie“

Auch hier ist wieder ein Feld „*Bezeichnung*“ gegeben, in welches neue Vertragskategorien eingefügt und durch den Link „**Vertragsart hinzufügen**“ in den Datenbestand mit aufgenommen werden kann. Die Liste der Stammdaten der Vertragskategorie wird dann um diese Vertragsart vergrößert.
Soll aus eben dieser Liste eine Vertragsart entfernt werden, ist diese anzuklicken. Das Feld „*Bezeichnung*“ wird folglich automatisch ausgefüllt und durch den Link „**Vertragsart entfernen**“ wird die Liste der Stammdaten der Vertragskategorie um diese Vertragsart verringert und die Vertragsart aus dem Datenbestand entfernt.
Wenn die Felder leer gesetzt werden sollen, ist der Link „**Abbrechen**“ zu nutzen. Wenn die Vertragspoolseite aufgerufen werden soll, steht der Link „**Start**“ zur Verfügung.
Nach dem Schema dieser Seite sind auch alle folgenden Seiten der Stammdatenpflege aufgebaut. Hierzu gehören noch die Seiten mit den Stammdaten zur „**LVM-Klasse**“, zu dem „**Betriebssystem**“, zum „**Hersteller**“, zur „**Sprache**“, zur „**Schlüsselkategorie**“ und zum „**Beschaffungsweg**“. All diese Seiten bieten ebenfalls die Möglichkeit, neue Daten zu den jeweiligen Stammdaten hinzuzufügen oder welche von dort zu entfernen.
Dabei können Daten nur entfernt werden, wenn sie *nicht mehr in Gebrauch* sind. Das Entfernen wird also verweigert, wenn beispielsweise noch in einem Vertrag die zu entfernenden Daten genutzt werden und so vergeben sind. Dadurch soll wiederum verhindert werden, dass die Daten inkonsistent werden.

7. Archiv

Die Seite „**Abgelaufene Verträge**“ kann im License-Management-System auf jeder Seite durch den Link „**Archiv**“ direkt erreicht werden. Diese Seite dokumentiert alle Verträge, deren Laufzeit abgelaufen ist. Das ist auch direkt durch die roten Ampeln klar ersichtlich. Der Gebrauch diese Daten ist demnach nicht mehr möglich, ohne damit eine illegale Handlung zu vollziehen. Sie werden aber dennoch gespeichert, damit bei zukünftigen Vertragsverhandlungen bessere Konditionen erreicht werden können, weil beispielsweise bereits Vorgänger der neuen Programme gekauft und in Gebrauch waren. Von dieser Seite führt der Link „**Start**“ direkt auf die Vertragspoolseite, damit ein direkter Vergleich der noch nutzbaren Daten mit den bereits abgelaufenen Daten möglich ist. Soll ein Vertrag, der bereits im Archiv ist, reaktiviert werden, weil beispielsweise ein Renewalvertrag abgeschlossen worden ist, sind die im Archiv gespeicherten Daten bei „**Vertrag hinzufügen**“ wieder einzugeben, auch mit dem alten Beschaffungsdatum. Diese Daten müssen dann wieder aus dem „**Archiv**“ entfernt werden. Dies geschieht, indem die individuelle Sachnummer des Vertrags angeklickt wird, woraufhin ein Fenster erscheint, in dem gefragt wird, ob der Vertrag wirklich entfernt werden soll.

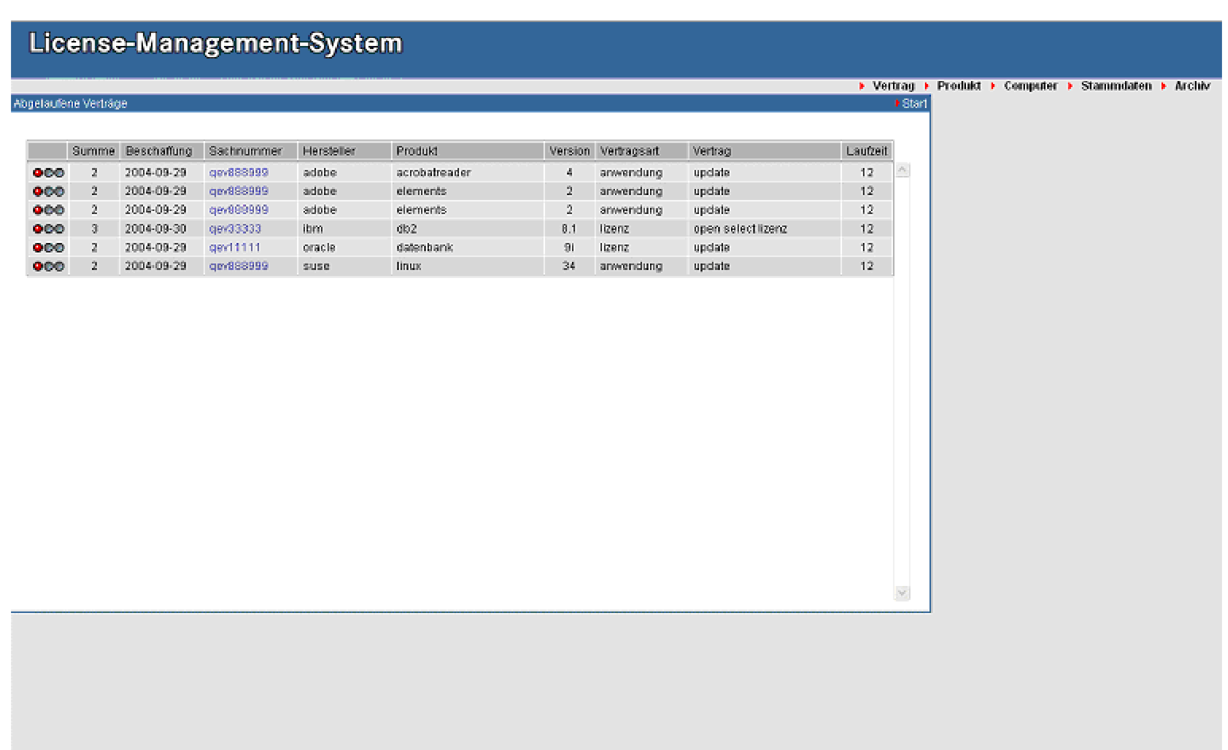


Abbildung: die Seite „Abgelaufene Verträge“ stellt die Verträge dar, welche nicht mehr im aktiven Datenbestand benutzt werden